INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

IDENTIFICATION OF REDUNDANT FAULTS

by

HONGDU ZHAO

A thesis submitted to the School of Graduate Studies
and Research of the University of Ottawa
in partial fulfilment of the requirements of
Master of Science in Computer Science*

January 1997

*The MCS Program is a joint program with
Carleton University, administered by the
Ottawa Carleton Institute of Computer Science

Acknowledgement

I would like to take this opportunity to express my appreciation to Professor Birta of the University of Ottawa who supervised me in the preparation of this thesis and Professor Abou-Rabia of Laurentian University who introduced me to my research topic. Their overwhelming support, guidance and encouragement have ensured the completion of this thesis.

Abstract

This thesis presents a redundancy identification (RI) method, namely, the Improved Structure Based (ISB) method, which identifies redundant faults in logic circuits. Redundant faults are those faults that are found not detectable by any test pattern in the process of automatic test pattern generation (ATPG) because they do not affect the normal circuit function. However, valuable time is wasted in ATPG algorithms due to the existence of redundant faults. The main purpose of ISB method is to locate redundant faults prior to applying ATPG algorithms. The ISB method does RI without search as opposed to most existing ATPG algorithms. The ISB method also presents significant improvement over existing structure-based methods for RI. The proposed ISB method can be applied efficiently to logic circuits with multiple-stem regions and multiple regions.

Figure List

Figure List

List of Tables

TABLE OF CONTENTS

CHAPTER I
INTRODUCTION

1.1 Overview of the Problem to Be Studied

Modern technology has made possible the fabrication of logic circuits of ever increasing complexity. In spite of the complex functions which they support, the physical size of these circuits continues to get smaller. Furthermore their cost has consistently decreased. Such complex and very dense logic circuits are called Very Large Scale Integration (VLSI) and it is these circuits that provide the foundation for a broad spectrum of consumer and technological devices ranging from radios and television sets to automobiles and automated factories.

Logical circuits however are subject to faults that can arise either from manufacturing or design errors or from deterioration after extensive use (i.e., in a hostile environment). The difficulty of detecting faults clearly increases with the complexity of the circuit. Consequently the growing application of VLSI circuits has given rise to the need for better logic circuit testing methods and tools.

Logic circuit testing is typically carried out by applying test signals to a set of inputs to the circuit and then examining the response signals that result at a set of outputs of the circuit. This approach to testing is usually very time consuming, especially when there are many inputs to the circuit.

Another source of testing complexity is the occurrence of redundant faults. Such a fault is distinctive because, in spite of its existence, the circuit response to all possible inputs corresponds to the fault free case. It would therefore appear that such a fault is of no consequence because it does not influence the circuit's input/output behaviour. In a narrow sense this is correct but it must be understood that such faults can undermine the detection of other faults in the circuit(which will be introduced in the following chapter). Therefore it is crucial to have methods to detect points in the circuit where such faults could exist.

Many attempts have been made in the past to reduce the cost of redundant fault identification. One of the most attractive breakthroughs was the one presented by Mohan Harihara and P. R. Menon in 1989. Their approach for identifying redundant faults depends exclusively on analyzing the structure of the logic circuit; i.e., it eliminates the need for any testing activity.

The work outlined in this thesis is based on that of Harihara and Menon and, more specifically, it extends the structure-based ideas for identifying redundant faults.

1.2 Contribution of This Thesis

As noted above, the work in this thesis focuses on the structure-based ideas of Harihara and Menon for locating redundant faults. The specific results that have been obtained are as follows:

1. An improved (simplified) procedure for "controlling-value" path* identification has been developed.

2. A set of equivalence laws has been formulated. These laws are applied after a set of "initial redundancies" is identified. This approach significantly enhances the efficiency of the redundancy identification.

3. Our structure-based method is applicable to a broader class of combinational circuits than the Harihara and Menon's method as described in [23].

---

* Terminology within quotes will be elaborated in later sections.

1.3 Organization of This Thesis

Following the introductory remarks in this first chapter, we give in Chapter 2, an introduction to logic circuit testing and redundant faults. In addition, most of the terminology that is used in the thesis is defined and explained with examples. Chapter 3 introduces a number of existing logic circuit fault testing and redundant fault identification methods. Mohan Harihara and P. R. Menon's method is introduced in detail.

The notion of equivalent faults is fundamental to the contribution of our work. This notion is presented in Chapter 4 together with several important Equivalence Laws. Chapter 5 presents our new approach for the detection of redundant faults and compares the approach with other methods.

A summary of the work achieved and our conclusions are provided in Chapter 6.

CHAPTER II
LOGIC CIRCUIT TESTING

2.1 Automatic Test Pattern Generation

Logic circuits are constructed by interconnecting elements called gates whose inputs and outputs are restricted to two values typically denoted by 0 and 1[16]. For example, Fig. 2.1.1 is a logic circuit with three gates g1, g2 and g3, the gates are connected with ten lines marked as a, b, ... etc. Note, in particular, here that there are three different lines associated with the junction point s1; namely, z, m and n. Line a and line b are input lines of gate g1, line k is the output line of g1 and an input line of g3 as well.

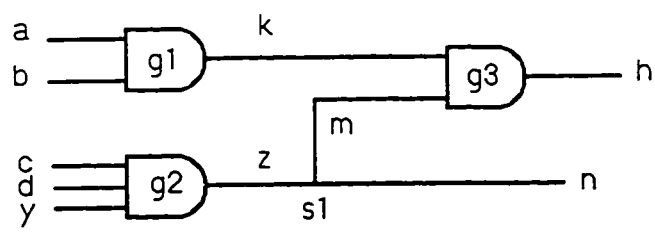


Fig 2.1.1

The external input lines in a circuit are defined as the **primary input lines** of the circuit (such as line a, b, c, d, y in Fig. 2.1.1), while the external output lines are defined as the **primary output lines** (Such as line h, n in Fig. 2.1.1). We define the direction from primary input lines to primary output lines as the **circuit direction** of the circuit.

When a line branches out into several lines at a point s in the circuit, that point s is called a **stem** and the line is called the **stem line** of s. In Fig. 2.1.1, the point s1 is a stem because line z branches out into line m and line n (line z is the stemline of s1).

There are many types of gates in logic circuits. In our considerations we deal only with AND, NAND, OR, NOR and NOT gates, which are the most fundamental types of gates. Each of these gates has only one output line. Except for the NOT gate, which has only one input line, all other gates have more than one input lines.

Figure 2.1.2 shows the five types of gates. The figure indicates how each gate operates and this operational behaviour is formalized using a truth table.

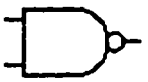
GATE TYPE	SYMBOL	OPERATION	TRUTH TABLE

AND

Output = 1 if all input = 1. Otherwise output = 0.

Input Assignments	Output
00	0
01	0
10	0
11	1

NAND



Opposite to the AND gate.

Input Assignments	Output
00	1
01	1
10	1
11	0

OR

Output = 1 if any input = 1. Otherwise output = 0.

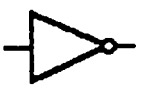
Input Assignments	Output
00	0
01	1
10	1
11	1

NOR

Opposite to the OR gate.

Input Assignments	Output
00	1
01	0
10	0
11	0

NOT



Output is always opposite to the input.

Input Assignments	Output
0	1
1	0

Table 2.1.2 Truth Table

In Fig. 2.1.2, it should be noticed that for each of the gates, there is a value which, if applied to any one of the input lines, uniquely determines the value of the output. The value in question is called the **controlling-value (cv)** of the gate. The following table shows the controlling-value of each gate type.

gate type	controlling value	output
AND	0	0
NAND	0	1
OR	1	1
NOR	1	0
NOT	both 0 and 1	the opposite value

Table 2.1.1 Summary of Controlling-values

The NAND, NOR and NOT gates are considered having an **inversion** function, because their output values are always opposite to the controlling values on their input lines.

A **logical fault** is a failure in any part of the circuit that causes its behaviour to differ from the fault free circuit. A **stuck-at fault** is a particular type of logical fault where a signal line is permanently fixed at a logical value (i.e. 0 or 1). In this thesis our concern is exclusively with stuck-at faults and whenever the word 'fault' is used, a stuck-at fault is implied.

There are only two possible kinds of stuck-at faults; namely, stuck-at zero faults and stuck-at one faults. These are denoted as **s-a-0** and **s-a-1** respectively.

Figure 2.1.3(a) shows a fault free AND gate while Figure 2.1.3(b) shows an AND gate with a s-a-1 fault. The truth table for the output for each of these cases is given below:

a	b	f	f*
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

Fig. 2.1.3 (a)

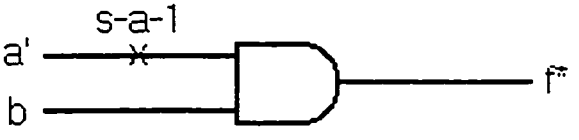


Fig. 2.1.3 (b)

In Fig. 2.1.3(b), the AND gate has two input lines and there are six possible single stuck-at faults; namely,

a: s-a-0 or s-a-1, b: s-a-0 or s-a-1, f: s-a-0 or s-a-1

In general, the total number of possible single stuck-at faults for a circuit with k input and output lines is 2*k.

In a logic circuit, a fault is called a **single fault** when it is the only fault in the circuit. If more than one fault exists in a circuit, then a **multiple fault** exists. Fig. 2.1.4 shows a multiple stuck-at fault in an AND gate.

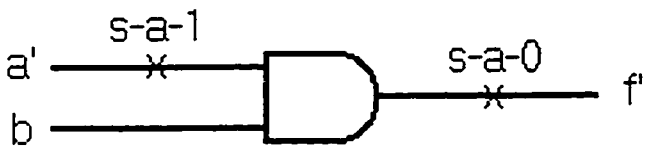


Fig. 2.1.4

If the possibility of multiple faults is allowed, the number of distinct fault configurations that could exist in a circuit with k lines, is $3^k - 1$.

The existence of a fault in a logic circuit can be established only through the information provided at the primary output lines. In general, the same erroneous value at a primary output may arise from any one of several possible stuck-at faults. This is illustrated in Fig 2.1.5. The two circuits shown provide the same erroneous output when the primary input is ab = 11(see the table in Fig. 2.5.1). A group of faults is said to be **equivalent** if each causes a circuit to manifest a malfunction in exactly the same way. So the single stuck-at fault (a:s-a-0) and the single stuck-at fault (f":s-a-0) in Figure 2.1.5 are equivalent.

a	b	f'	f''
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	0

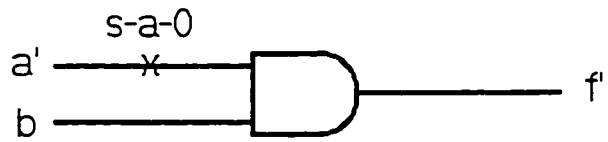


Fig. 2.1.5 (a)

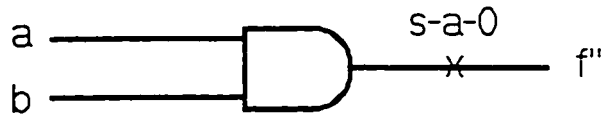


Fig. 2.1.5 (b)

Fault testing procedures are used to identify stuck-at faults in a logic circuit. In this process test patterns are applied at the primary input lines of the circuit and the circuit responses are compared with those of the corresponding fault free circuit. A **test pattern** is an assignment of values to the primary input lines which:

a) Strives to give the line where a fault is located to have a logical value which is opposite to the fault value (this process is called **fault sensitizing**).

b) Propagates the effect of the fault from the faulty site (the line where the fault is located) to a primary output (this process is called **fault propagation**, the path on which the fault is propagated is called a **sensitized path**).

Fig. 2.1.6 gives an example of a test pattern to test the s-a-0 fault on line a. Because of this fault the primary output of the circuit is always 0. The fault sensitizing is realized by assigning a value of 1 to line a, which is opposite to the fault value. The fault propagation is realized by assigning a value of 1 to line b. With this particular primary input value, the primary output should be 1; however, it remains at 0. This

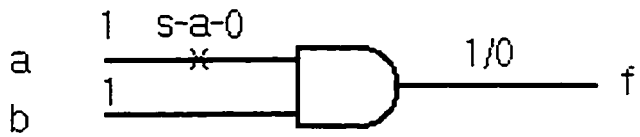


Fig.2.1.6

reveals the existence of the fault.

Alternately consider Figure 2.1.7 where a s-a-1 fault exists on line a. Now fault sensitizing is realized by assigning a value of 0 to line a. The fault propagation is realized by assigning the value of 1 to line b. The circuit output expected for this input is 0 but a value of 1 will occur.

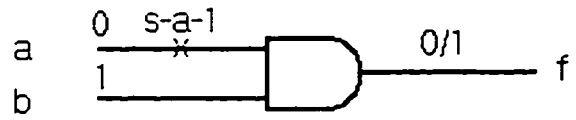


Fig.2.1.7

Note that we use the notation 1/0 to indicate that 1 is the fault free value and 0 is the fault value on a line. This notation will be used throughout this paper (the notation 0/1 is equivalently interpreted).

A sensitized path may include many gates and some of these gates may have inputs that are not on the sensitized path (these are defined in the following section to be; "off-path inputs"). All such inputs must be assigned their non-controlling values to ensure the propagation of the fault. This is obvious because if any off-path input line of a gate on the sensitized path has been assigned a controlling value, the output value of that gate will be uniquely decided, this blocks the fault propagation.

Now it is easy to understand that the process to generate test patterns automatically for VLSI circuits are called **Automatic Test Pattern Generation (ATPG)**.

2.2 Some Concepts, Terminology and Definitions

Some basic concepts about logic circuit testing have been introduced in the previous sections. In this section we introduce some additional concepts that are specifically related to the redundant fault identification problem.

A number of important concepts and definitions pertinent to the discussions in this thesis are introduced below and illustrated using Fig. 2.2.1(a).

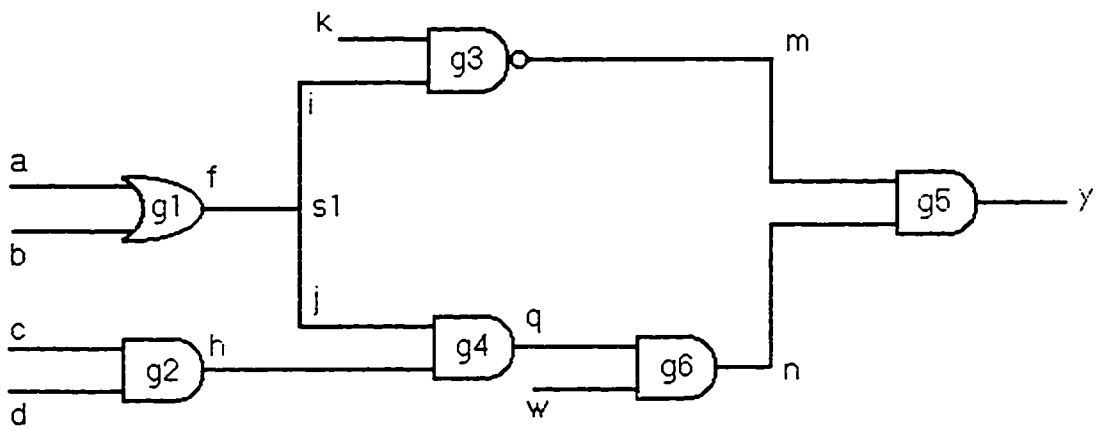


Fig. 2.2.1(a)

Reconvergent gate: A gate G_r is a reconvergent gate of a stem s, if there exist two or more disjoint paths from s to G_r[23].

For example, g5 in Fig 2.2.1(a) is a reconvergent gate of stem s1, because there are two disjoint paths from s1 to g5.

We frequently have the need to consider paths between a stem and its reconvergent gate. It will be our convention not to include either the stem or the reconvergent gate in such a path specification. Such paths will be called **sG-paths**. In Figure 2.2.1(a), i - g3 - m and j - g4 - q - g6 - n are both sG-paths.

When a sG-path P starts at line L1 and ends at line L2, we denote the sG-path as **P(L1, L2)**. Thus, in Fig. 2.2.1(a), we have P(i, m) and P(j, n).

Let g be a gate on a sG-path, the input line to gate g which is on the sG-path is called **on-path (OP) input line** of g, all other input lines to g are called **off-path (FP) input lines**.

For gate g3 which is on the sG-path P(i, m), line i is the OP input line of g3 and line k is an FP input line of g3.

A reconvergent region **R(s, G)*** is a related concept. A reconvergent region R consists of the following four parts:

---

* In reference[23], it is denoted by (G, s).

a) the stem s and its stem line.

b) the reconvergent gate G and its output line.

c) all the sG-paths from s to G.

d) the FP input lines of all gates on all sG-paths from s to G.

In Fig. 2.2.1(a), the reconvergent region R(s1, g5) is:

{s1, f, g5, y, i, m, j, q, n, k, h and w}.

In a region R(s, G), all gates which are directly connected to s are called **first-gate-level (FGL) gates**. The set of such gates is denoted by **g'(R)**. The set of all OP input lines which belong to the gates in g'(R) is denoted by **I(g'(R))**.

In the region R(s1, g5) in Fig. 2.2.1(a); e.g.,

g'(R) = {g3, g4}.

I(g'(R)) = {i, j}.

Now, let us assume we are testing the single stuck at fault (i: s-a-0). The fault sensitization requires that we assign value 1 to line i to sensitize the fault (see Fig. 2.2.1(b)).

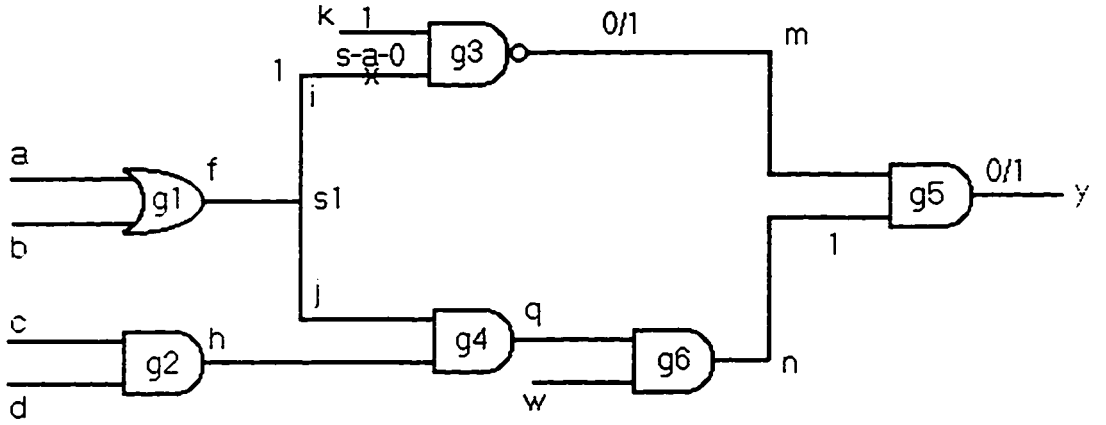


Fig. 2.2.1(b)

Because line i is on P(i, m), a "backward tracing" process has to be carried out to establish values on the primary input lines which will yield the desired values for line i. The process that traces backward from values on internal lines to values on the primary input lines is called line **justification**. With respect to our example in Fig. 2.2.1(b) we achieve line justification using the following steps:

. $i = 1 \Rightarrow f = 1 \Rightarrow ab = 01$ (or $ab = 10$, or $ab = 11$)

The fault propagation requirement can be carried out by assigning $k = 1$ and $n = 1$. Because line n is P(j, n), a line justification must again be carried out as follows:

. $n = 1 \Rightarrow q = 1$ and $w = 1$.
. $q = 1$ implies:
- $j = 1 \Rightarrow f = 1$ (the line justification for $j = 1$ can stop here because $f = 1$ has already been established in the line justification for $i = 1$.)
- $h = 1 \Rightarrow c = 1$ and $d = 1$.

So a test pattern is abcdkw = 011111. In fact abcdkw = 101111 and abcdkw = 111111 are also test patterns for the fault (i: s-a-0). Which one is chosen depends on the algorithm used in the implementation of ATPG.

Now consider the single stuck-at fault (n: s-a-0) in Fig. 2.2.1(c). Assign n = 1 to sensitize the fault, m = 1 to propagate the fault. Line justifications are necessary for these two assignments because neither is a primary input.

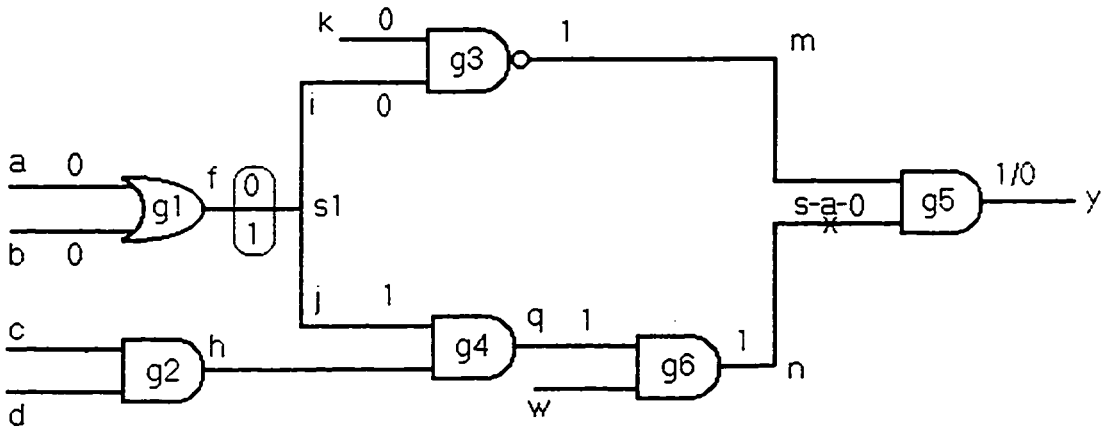


Fig. 2.2.1(c)

Line justification for m = 1 can be carried out in several ways. One is as follows:

- k = 0.
- i = 0 ➡ f = 0 ➡ a = 0 and b = 0.

Line justification for n = 1 is as follows:

- q = 1 implies:
  - j = 1 ➡ f = 1.

The process cannot be continued because a conflict has happened at line f. When a conflict occurs between the current assignment and some previous assignment during a line justification process, it is required to return to the previous assignment(s) and make some changes. This process is called **backtracking.**

The backtracking process can be carried out in several ways and we illustrates one of these below:

We first summarize the assignments that have been made (ignoring the final assignment, f = 1, which triggered the conflict). These assignments are listed below in the reverse order in which they were made:

(j = 1, q = 1, a = 0 and b = 0, f = 0, i = 0, k = 0)

Altering either j or q would result in an unacceptable value of 0 for n; there neither j nor q can be changed. A change in line a from 0 to 1 result in f = 1 which provides the line justification for n = 1. It must now be confirmed that the necessary value of m = 1 can still be achieved. This is clearly possible because f = 1 ➡ i = 1 but m still remains of the desired value of 1 (which is the necessary condition for the propagation of the fault (n: s-a-0)), we have thus resolved the conflict problem.

Let g be a gate in a circuit. Generally there are paths from g to the primary outputs. If any path from g to a primary output must pass through a gate $g^{\cdot} \neq g$, then g is said to be **bound** by $g^{\cdot}$ and g^{*} is called a **dominator** of g. Similarly if every path from g to a primary output must pass over a line L, then g is said to be bound by $L^{\cdot}$ and $L^{\cdot}$ is a dominator of g.

In a similar way a line L can be bound by a gate $g^{\cdot}$ or a line $L^{\cdot}$, then $g^{\cdot}$ or $L^{\cdot}$ is a dominator of L.

In Fig. 2.2.1(c), lines a and b are both bound by line f. All gates in the circuit are bound by line y (or line y is a dominator of all gates) and all gates (apart from g5) are bound by g5.

In a reconvergent region R(s, G), a gate g or a line L on a sG-path is often bound by the reconvergent gate G because a path from g or L to any primary output must pass through G (see gate g4 or line q in Fig. 2.2.1(c)). However, it can occur that a stem t exists and a path, P1, from this stem point branches out to a primary output without passing through a reconvergent region. In such a situation the path between the stem point s and the stem point t is no longer bound by the reconvergent gate, G. Furthermore, we call the path P1 an **exit path** relative to the sG-path under consideration.

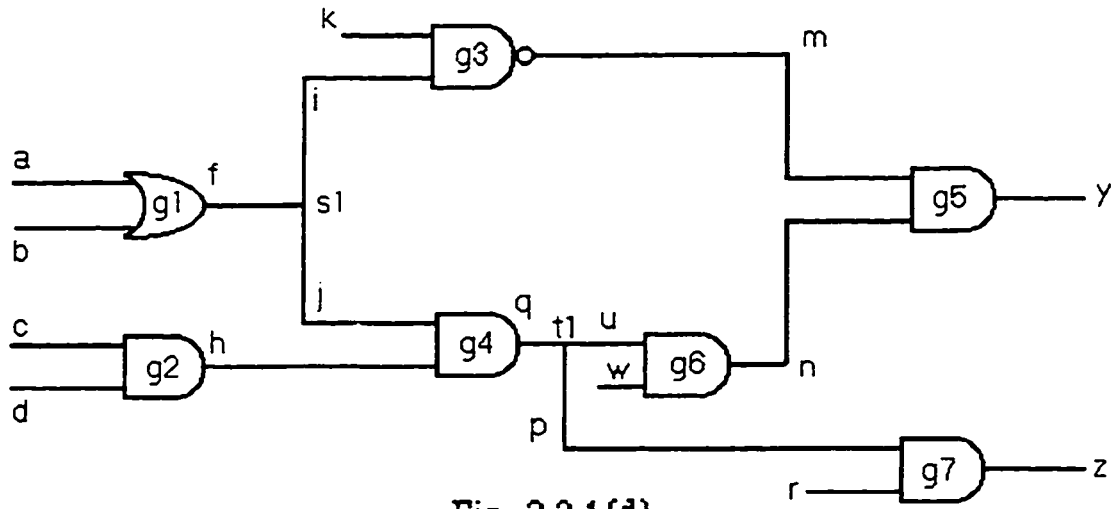


Fig. 2.2.1(d)

For example, the path t1 → p → g7 → z in Fig. 2.2.1(d) is an exit path of the sG-path P(j, n) because it branches out from the sG-path P(j, n) and goes to an primary output line z. Because the existence of this exit path, line j, gate g4 and line q are no longer bound by the reconvergent gate of g5 (Note between that line u, g6 and line n are still bound by g5).

If a region consists of m stems (m > 1), and all stems reconverge to a common reconvergent gate, we call such a region a **multiple-stem region.**

2.3 Redundant Fault and Redundancy Identification

Consider the fault (d: s-a-1) in Fig. 2.3.1(a).

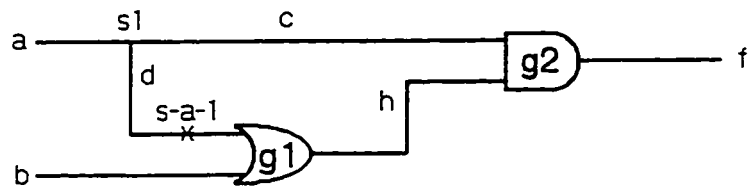


Fig. 2.3.1 (a)

To test the fault, we should assign a = 0 to sensitize it and assign b = 0 to propagate it. Thus ab = 00 is the only possible pattern for testing this particular fault. The effect of applying this test pattern to the circuit is shown in Fig. 2.3.1 (b).

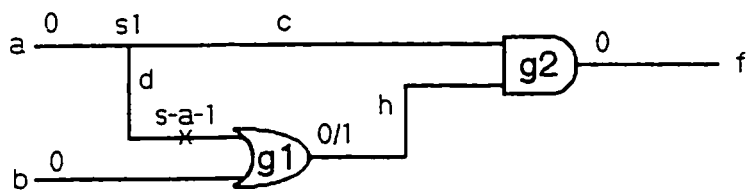


Fig. 2.3.1 (b)

To propagate the fault from line h to line f, the assignment c = 1 is a necessary condition. But a = 0 implies c = 0 which blocks the propagation of the fault since c = 0 forces the output of g2 to 0.

On the other hand, since c = 0 implies f = 0, it appears that the faulty circuit has the same response as the fault free circuit to input ab = 00. This means that the input pattern ab = 00 required for testing (d: s-a-1) is not a test pattern. We have, therefore, following important observation:

This fault does not change the function of the circuit as shown in the following truth table and consequently there is no test pattern for this fault.

a	b	c	d	h	f with (d: s-a-1)	f when fault free
0	0	0	0/1	0/1	0	0
0	1	0	0/1	1	0	0
1	0	1	1	1	1	1
1	1	1	1	1	1	1

Table 2.3.1 Truth Table

We call a single stuck-at fault a **redundant fault** or a **redundancy** if there is no test pattern that can be generated for the fault testing or if the fault does not change the function of the circuit. The fault (d: s-a-1) in Fig. 2.3.1(a) provides an example of a redundant fault.

In some special cases, both s-a-0 and s-a-1 faults on a line L do not change the function of the circuit and hence none of them has a test pattern; we denote this by **(L: s-a-β).**

Some redundancies can be deduced from the existence of other redundant faults. In Fig. 2.3.1(c), for example, assume the fault (d: s-a-1) is redundant, and assume also the fault (h: s-a-1).

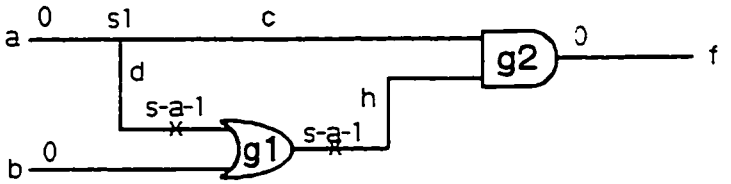


Fig. 2.3.1 (c)

The requirement to sensitize (h: s-a-1) is b = 0 and d = 0. Because of the fault (d: s-a-1), the requirement d = 0 cannot be realized, hence (h: s-a-1) is also a redundant fault.

If the existence of a redundancy A provides a sufficient condition for the existence of another redundancy B, we call the redundancy B an **equivalent redundancy** (or an **equivalent redundant fault**) of A.

In Fig. 2.3.1(c), for example, (h: s-a-1) is an equivalent redundant fault of (d: s-a-1).

The existence of a redundant fault does not change the normal operation of a circuit. In a circuit, there can be several lines where a redundant fault could exist. However, it is important to identify these lines and to eliminate the redundant fault for several reasons*:

First, redundant faults can block the propagation of some other sensitized single stuck-at faults. For example, in Fig. 2.3.2(a), assume line m has a s-a-1 fault. In order to propagate the fault, the FP-input line k to g2 must be assigned the non-controlling value of 1. But such an assignment to line k is not possible because it conflicts with the assignment d = 0 that is necessary to sensitize the fault. Thus, this fault propagation is blocked at g2. Furthermore, any single stuck-at fault in line b, c, p, q, h and at any line in the **bounded area** such as b_1, b_2, ..., b_n cannot be propagated to line L. (An **area** of a given circuit is a portion of the circuit which can be contained within a closed contour superimposed on the circuit. An area is **bounded** if every gate and every line within it is bounded by some gate or some line outside of the area.)

---

* There are specific cases where redundant faults are introduced for some particular reasons. We do not consider such cases.

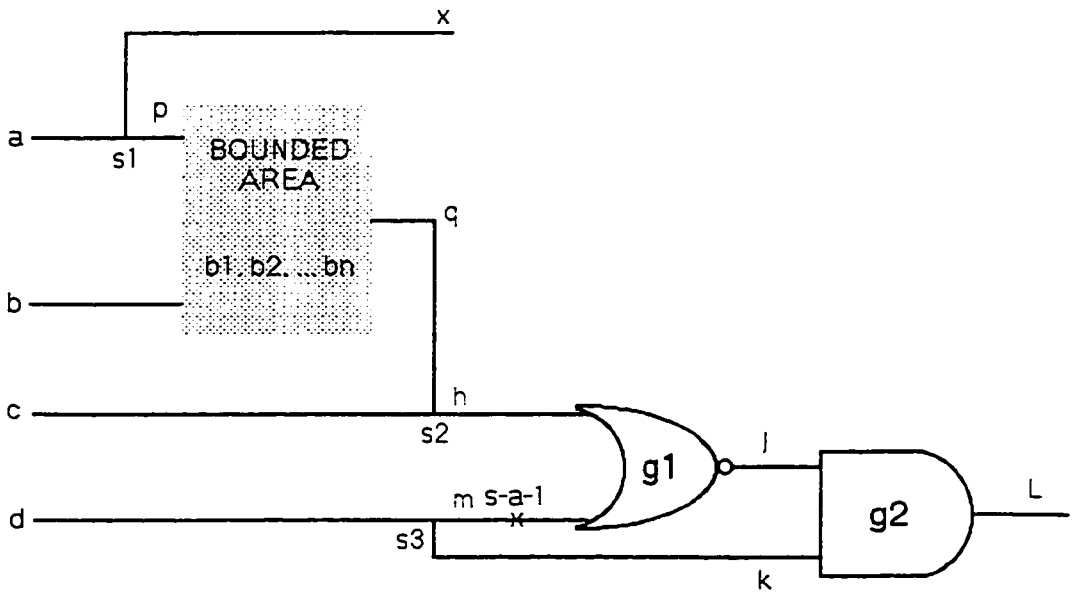


Fig. 2.3.2(a)

Secondly, redundant fault may mask some detectable faults. For example, the detectable fault s-a-1 at line k in Fig. 2.3.2(b) can be sensitized and propagated by the test pattern cd = 00 which results in an output L = 1 which is faulty. However if the redundant fault (m: s-a-1) occurs then j is fixed at logical value 0, which implies the primary output L = 0. Therefore the fault (k: s-a-1) is masked by the redundant fault because L = 0 is the response of a fault free circuit to the test pattern cd = 00.

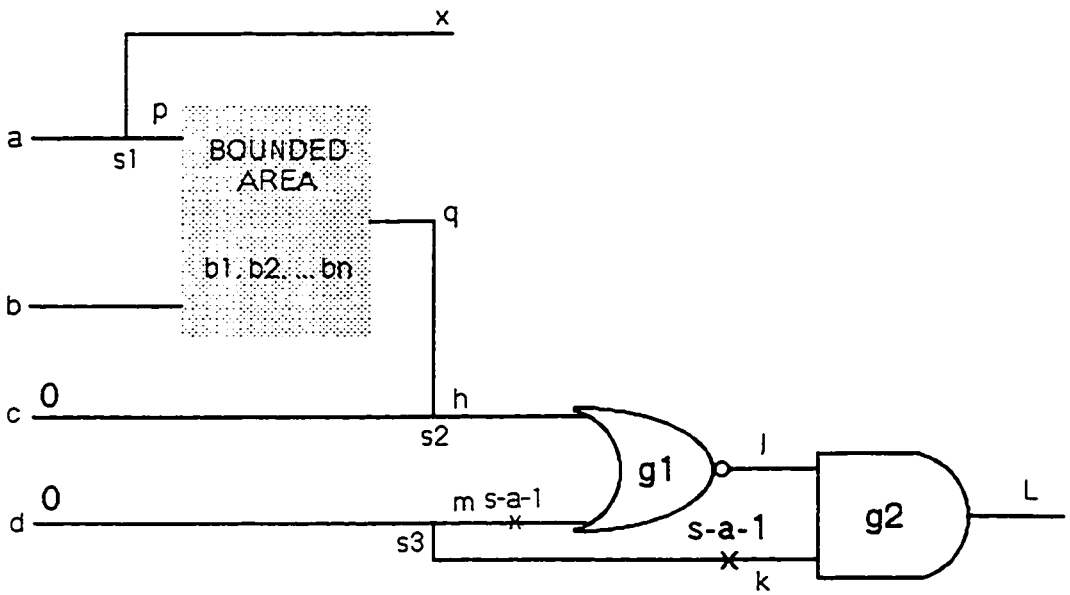


Fig. 2.3.2(b)

Finally, redundant faults make the test pattern generation procedure very time consuming. The reason is that when a stuck-at fault has not been identified redundant at the beginning of the test pattern generation, the test pattern generation procedure typically conducts an exhaustive search trying to generate a test pattern for this fault. This exhaustive search ends with the conclusion that the fault is redundant.

The process of identifying redundant faults (or **redundancies**) in VLSI circuits is called **Redundancy Identification (RI)**.

CHAPTER III
OVERVIEW OF THE EXISTING ATPG AND RI METHODS

There are relatively few methods specially for RI in VLSI circuits. However, RI is involved in every ATPG method. In this chapter we briefly review some ATPG and RI methods.

3.1 Some Early Approaches

3.1.1 D-ALGORITHM[16]

The D-algorithm is the first algorithm that comprehensively treated the ATPG problem. It is called the D-algorithm because it uses the letter D to denote 1/0 which is intended to indicate that the value on a line is binary one in a good circuit, but zero in faulty circuit. In a similar way, D represents 0/1.

The D-algorithm undertakes to implement the fault-sensitization, fault propagation concept(see section 2.1). When a single stuck-at fault is tested, the D-algorithm first marks the fault as D if the fault is a s-a-0 fault or as D if it is a s-a-1 fault. A sensitized path must now be built-up in order to propagate the fault to a primary output. Whenever necessary, a line justification process must be carried out and if a conflict occurs then a backtracking procedure is undertaken. If a successful fault propagation is achieved, then a test pattern for the fault is generated. If backtracking is carried out and an inconsistency still exists; i.e. there is no test pattern for the fault in question, then the fault is recognized as a redundancy.

For example, consider the fault (i: s-a-1) in Fig. 3.1.1. Its sensitization requires i = 0 (hence f = 0) and its propagation to the primary output at line y requires k = 1 and n = 1. Line justification for n = 1 requires that t = 1, q = 1, j = 1, h = 1. But note that j = 1 implies f = 1 which conflicts with the sensitization requirement of f = 0. There are no available options to consider hence backtracking is not feasible. Thus, the fault is identified as a redundant fault.

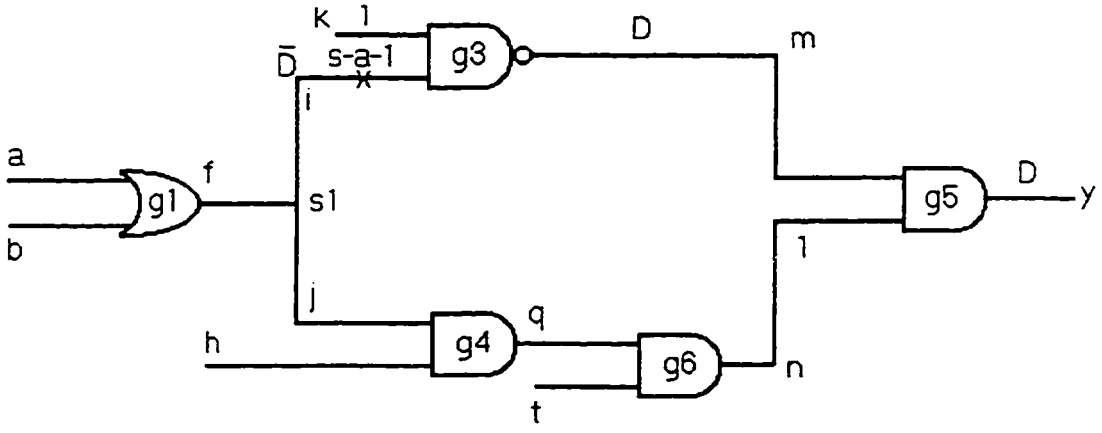


Fig. 3.1.1

PODEM (Path Oriented DEcision Making) Algorithm [16]

Like the D-algorithm, the PODEM approach focuses on the need to achieve the necessary line assignments arising from sensitization and propagation requirements. Rather than inferring the necessary input values by working backwards (as in the D_algorithm), PODEM directly makes primary input assignments and simply monitors their success in achieving the necessary internal values. A strategy for successively choosing the primary input assignments is an important part of the algorithm and depends on a backtracking process. A fault in a circuit is identified redundant if a test pattern cannot be found after exhaustive assignment at the primary input lines of the circuit.

Consider again the (i: s-a-1) fault in Fig. 3.1.1. Recall that sensitization requires that i = 1 and propagation requires n = 1 and k = 1. The PODEM algorithm proceeds by trying to find an assignment for the primary inputs (a, b, h, k, t) which yields the desired values for i, n and k. In this example, no such assignment exists hence the algorithm concludes that the fault is redundant.

FAN (FANout-oriented test generation) Algorithm [16]

PODEM reduced the number of backtracks because it only assigns logical values to primary inputs. But backtracking still has to be done when an assignment fails to generate a test pattern. The FAN algorithm further reduces the number of such operations by applying some strategies. We here summarize these as a collection of three rules:

Rule-1/3.1* The required internal values in the circuit often imply unique values at other lines within the circuit. Determine all such values.

Rule-2/3.1 Within a reconvergent region, when a line justification along one path reaches a stem line, postpone the line justification until the line justifications alone the other paths reach the same stem line.

Rule-3/3.1 When a contradictory requirement occurs at a stem line, assign a binary value to the line and see if the conflict can be solved by changing previous assignment(s). If the conflict still exists, assign the other binary value to the stem line and try again to change previous assignment(s). When both assignments at the stem line fail to solve the conflict, declare the fault as redundant right away and end the test pattern generation.

---

* Here and in the sequel, we use /"section#" to indicate in which section the items (rule, law, etc.) are introduced.

To illustrate the application of these rules, we use the sample circuit in Fig. 3.1.2(a).

Consider the fault (n: s-a-0). Sensitization requires that n = 1 and propagation requires that m = 1. Both lines m and n can be traced back to the stem line f; in other words, there is a path from line f to line m (path1) and from line f to line n(path2). Notice that n = 1 implies q = 1 and w = 1(This conclusion represents an application of Rule-1/3.1). The requirement for m = 1 does not generate internal assignments from Rule-1/3.1.

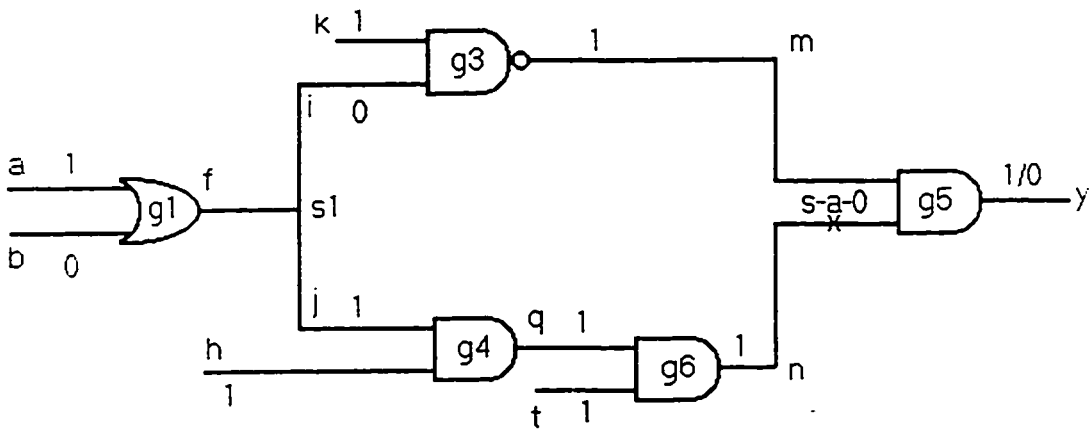


Fig. 3.1.2(a)

We choose one of the paths and carry out line justification; e.g., choose path1. The desired value of m = 1 can be achieved by several different assignments to the inputs of gate g3. We arbitrarily choose k = 1 and i = 0. Because the i = 1 assignment corresponds to a stem point, we now go to path2(application of Rule-2/3.1).

The q = 1 requirement can be achieved in several ways. One such assignment is j = 1 and h = 1. This however causes a conflict at the stem line f (as shown in Fig. 3.1.2(a)) and hence Rule-3/3.1 must be applied.

Assign f = 1 first. We find that the previous assignment k = 1 and i = 0 to achieve the requirement of m = 1 can be changed to k = 0 and i = 1 without violating the requirement, the conflict is thus resolved.

If we choose a = 1 and b = 0 to imply f = 1, the test pattern abhkw = 10101 is generated, shown as Fig. 3.1.2(b).

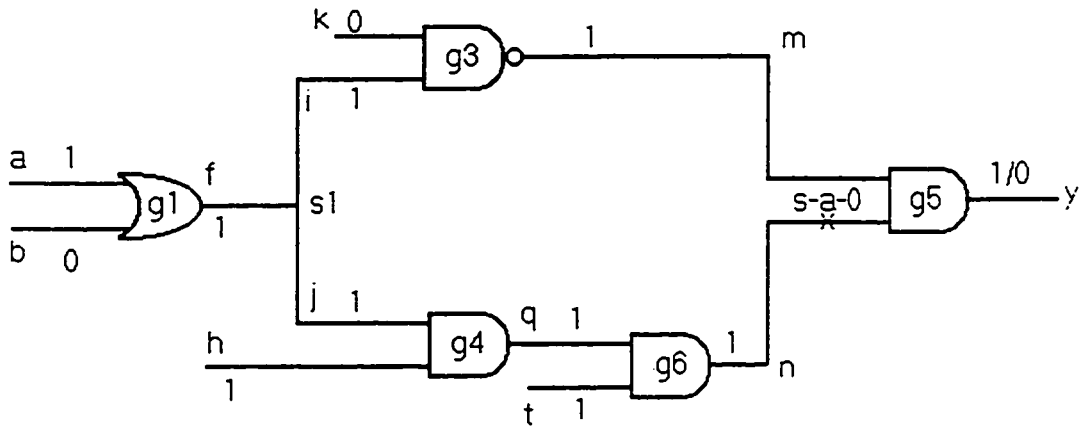


Fig. 3.1.2(b)

3.2 TOPS (TOPOLOGICAL SEARCH ALGORITHM)[27]

TOPS accelerates test pattern generation by reducing the search space and rapidly identifies many redundant faults even without line justification.

TOPS presents a combinational circuit as a directed acyclic graph(DAG). It represents gates as vertices and circuit lines as edges. For example, the circuit in Fig. 3.2.1 (a) is presented as a DAG shown in 3.2.1 (b).

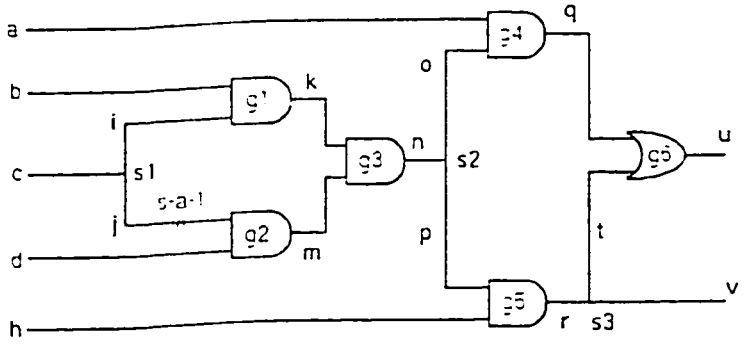


Fig. 3.2.1 (a)

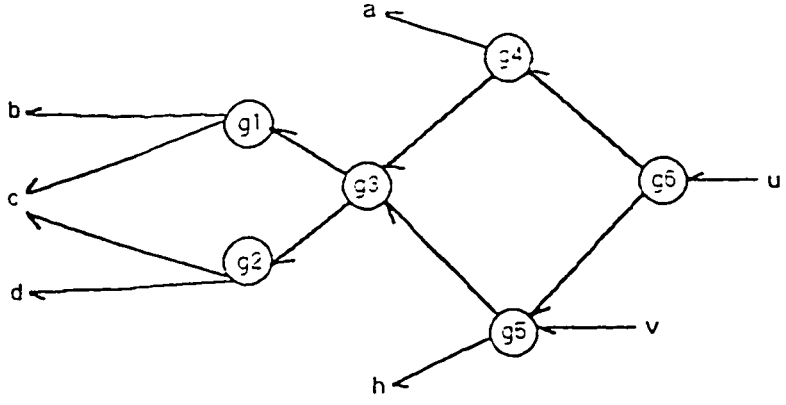


Fig. 3.2.1 (b)

This graph representation of a circuit plays an important role in the theoretical discussion within [27]. However, the underlying concepts of the technique can be described in terms of the given circuit and we adopt this approach.

The main contribution of the TOPS approach is the observation that a fault can be identified as redundant using the following procedure:

a) Find all gates which bound the line on which the target fault occurs, denote this set of gates by B.

b) If a conflict arises on any line which is directly connected to some element in B after the assignments for the fault sensitization and fault propagation (together with the consequences of these assignments), then the fault is redundant.

For example, the target fault (j: s-a-1) in Fig. 3.2.1(a) can be found to be redundant as follows:

a) Since line j is bound by g2 and g3 we have B = {g2, g3}.

b) The sensitization assignment j = 0 has many consequences. The only two that are of interest relate to line k and n since there are connected to g3 which is a member of B. Specifically j = 0 implies k = 0 and n = 0. Fault propagation requires d = 1 and k = 1 (other requirements of fault propagation are not relevant here because they do not relate to lines that are inputs to gates in B). A conflict has thus arisen on line k and we therefore conclude that the target fault (j: s-a-1) is redundant.

3.3 SOCRATES (Structure-Oriented Cost-Reducing Automatic TESt pattern generation system) [39]

SOCRATES is based on the FAN algorithm together with some procedures which enhance the generation of test patterns. Our introduction focuses on its search space concept and one of its procedures which plays the most important role in RI.

Search Space

A deterministic ATPG for a circuit with N primary inputs has a finite search space (2^N). This search space can be represented as an AVL tree (i.e. height balanced tree) with height N + 1. Fig. 3.3.1 is an AVL tree which represents the search space for any circuit having three primary inputs (denoted as a, b, and c), where each edge corresponds to a primary input and each node is the assigned value of the primary input associated with the edge that connects to the node above. Any path from the root node to a leaf node defines an input assignment. For a target fault, some of these paths are test patterns while others are not. The goal of test pattern generation is to determine those which are test patterns.

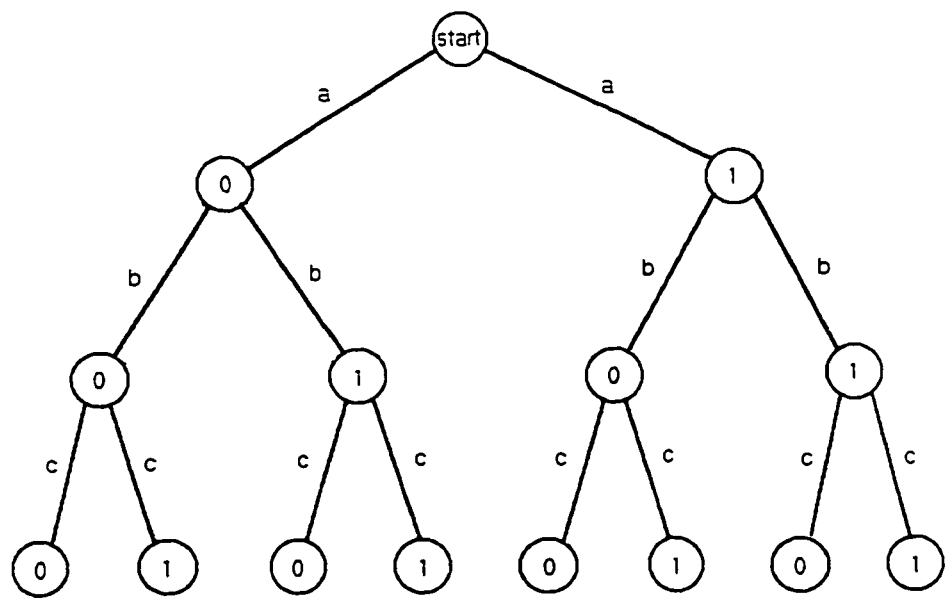


Fig. 3.3.1

With respect to any target fault, the search space for ATPG can be divided into two parts: the solution area and the non-solution area. We use the circuit in Fig. 3.3.2 for illustration.

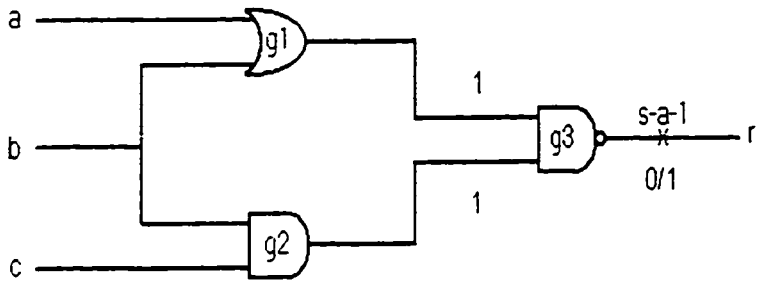


Fig. 3.3.2

Because the circuit in Fig. 3.3.2 has three primary inputs, its search space is as shown in Fig. 3.3.1. If we do an exhaustive assignment at these three inputs, two test patterns for the target fault (r: s-a-1) can be identified; namely, (a, b, c) = (0, 1, 1) and (a, b, c) = (1, 1, 1). The other assignments at the primary inputs a, b, c are not test patterns for the target fault because they all produce a binary 1 at line r which is not the required sensitizing value. The two categories of primary input assignment are represented in Fig. 3.3.3 using shaded and unshaded nodes. A test pattern is a path from the root to a leaf which does not traverse any unshaded node. Correspondingly, the set of unshaded nodes in Fig. 3.3.3 is called the solution area while the set of shaded nodes is the non-solution area.

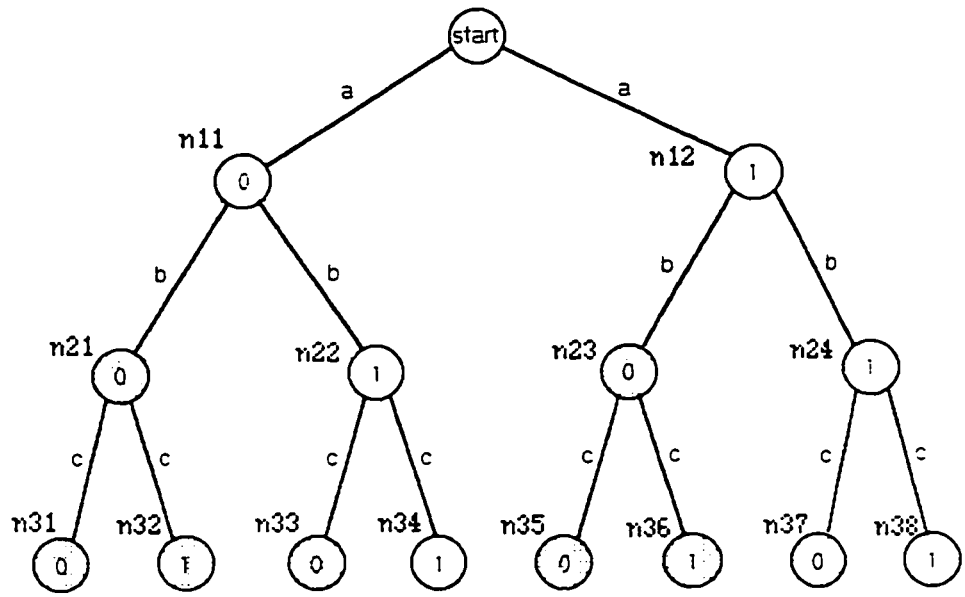


Fig. 3.3.3

A test pattern generation process can be considered as a traversal in the search space tree, when a path from the root node to a leaf node has been found in a solution area, a test pattern is generated; e.g., path n_{11}-n_{22}-n_{34} in Fig. 3.3.3 is a test pattern for the target fault (r: s-a-1) in Fig. 3.3.2. Note furthermore that any traversal procedure which enters the non-solution area, is futile and should be immediately abandoned in the interests of time-efficiency. Consequently, it is of great value to known as much of the non-solution area as possible, prior to beginning the graph traversal. In this regard, SOCRATES uses a procedure called "Improved Implication Procedure" as a pre-processor for ATPG.

Improved Implication Procedure

The Improved Implication Procedure is based on a logical identity called the **law of contraposition;** i.e., $(P \rightarrow Q) \rightarrow (Q \rightarrow P)$.

For example, in Fig. 3.3.2, $c = 0$ implies $r = 1$; i.e., $(c = 0) \rightarrow (r = 1)$. Application of the law of contraposition gives: $(r = 0) \rightarrow (c = 1)$. In other words, the desired value cannot be achieved with $c = 0$ and consequently, any node associated with the assignment $c = 0$ is within the non-solution area.

Similarly, we observe that $(b = 0) \rightarrow (r = 1)$. By the law of contraposition, $(r = 0) \rightarrow (b = 1)$, thus any node associated with the assignment $b = 0$ is within the non-solution area for the target fault (r: s-a-1). Furthermore, all subtrees of such nodes also fall within the non-solution area.

The Improved Implication Procedure corresponds to carrying out all possible applications of the law of contraposition. Then the test pattern generation can be carried out by examining the portions of the search space which have not been identified as belonging to the non-solution area. The target fault is redundant if the tree representation has no solution area.

3.4 EST (Equivalent STate hashing algorithm) [19]

The basis for EST can be developed using a Binary Decision Diagram (BDD). Fig. 3.4.1 is a sample circuit:

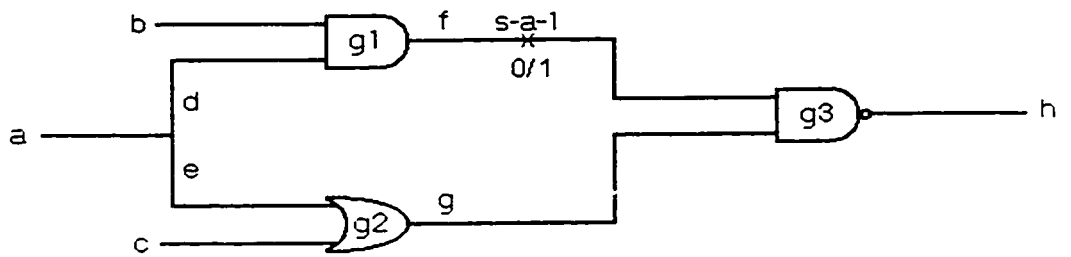


Fig. 3.4.1

A complete BDD is a binary tree which traverses the whole search space and shows all test patterns and all inconsistent assignments at primary inputs of the circuit which contains the target fault. The complete BDD for fault s-a-1 at line f is illustrated as Fig. 3.4.2.

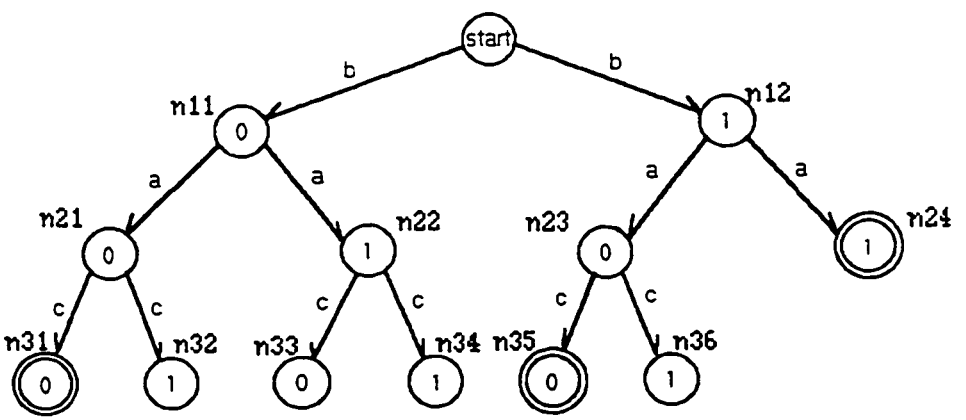


Fig. 3.4.2

In Fig. 3.4.2, paths n_{11}-n_{21}-n_{31}, n_{12}-n_{23}-n_{35} and n_{12}-n_{24} are all inconsistent assignments, whereas paths n_{11}-n_{21}-n_{32}, n_{11}-n_{22}-n_{33}, n_{11}-n_{22}-n_{34} and n_{12}-n_{23}-n_{36} are test patterns for the target fault. The nodes n_{31}, n_{35} and n_{24} have a double wall to indicate that they each correspond to an assignment on primary input line which gives rise to conflict. If all leaves have double walls, then the target fault is recognized as a redundant fault.

Any path from the root node to a leaf node which has length m and contains no double walled nodes is a test pattern (here m is the number of primary inputs to the circuit). EST uses a Depth-First-Search (**DFS**) process to search for such path.

Typically, the construction of a BDD tree for a circuit with a given target fault stops when a test pattern is generated. The specific test pattern that is generated will depend on the strategy used to construct the BDD tree. In Fig. 3.4.3, we show the case where DFS is used with the circuit given in Fig. 3.4.1.

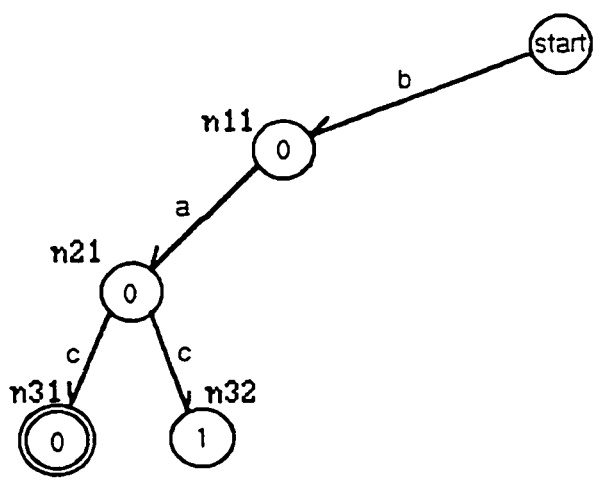


Fig. 3.4.3

An important contribution to RI which is embedded within EST is a redundancy identification method that is based on an equivalence rule. This rule can be stated as follows, which is called **Rule-A/3.4** in our later discussion:

Suppose L1 and L2 are two lines in a circuit with the property that a value i on line L1 implies a value j on line L2. Then the existence of the redundant fault (L1: s-a-i) implies the existence of the redundant fault (L2: s-a-j).

Therefore, during the test pattern generation, after some redundant fault are recognized, EST will apply Rule-A/3.4 to identify other unidentified redundant faults.

We demonstrate this method using an example introduced in [19]. This example is a part of the benchmark circuit c2670 given in [11] and is shown in Fig. 3.4.4(a).

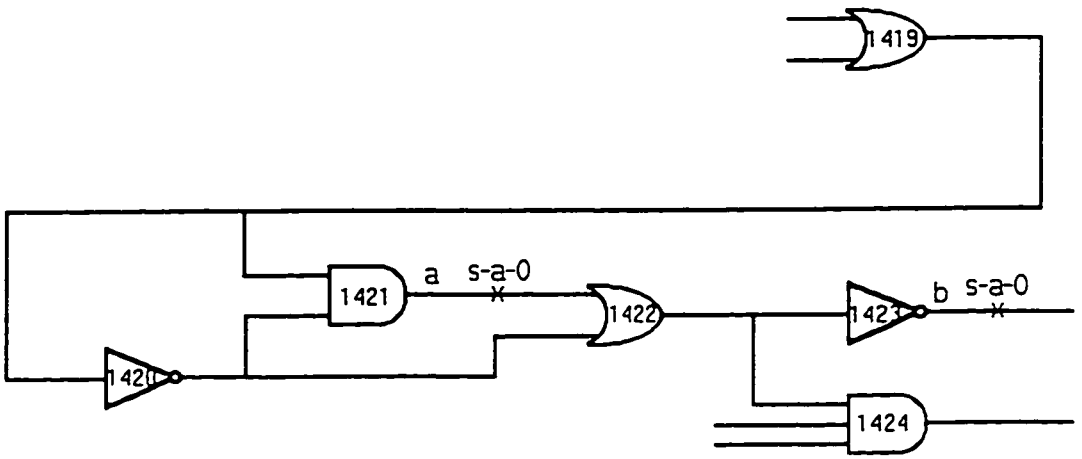


Fig. 3.4.4(a)

Suppose faults (1421: s-a-0) and (1423:s-a-0) have been found redundant. In Fig. 3.4.4(b), the circuit is redrawn with the stuck-at value shown on the lines.

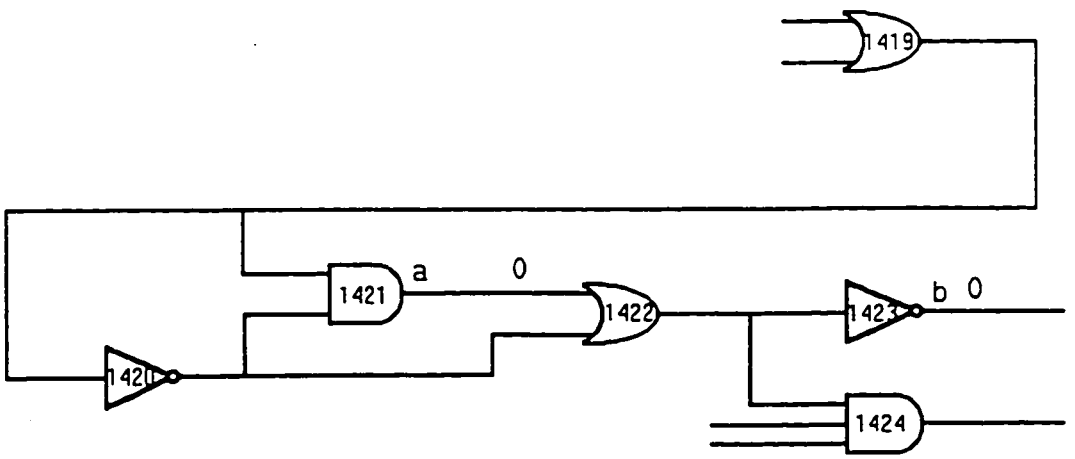


Fig. 3.4.4(b)

The implicit consequences of the assignments shown in Fig. 3.4.4(b) are shown in Fig. 3.4.4(c).

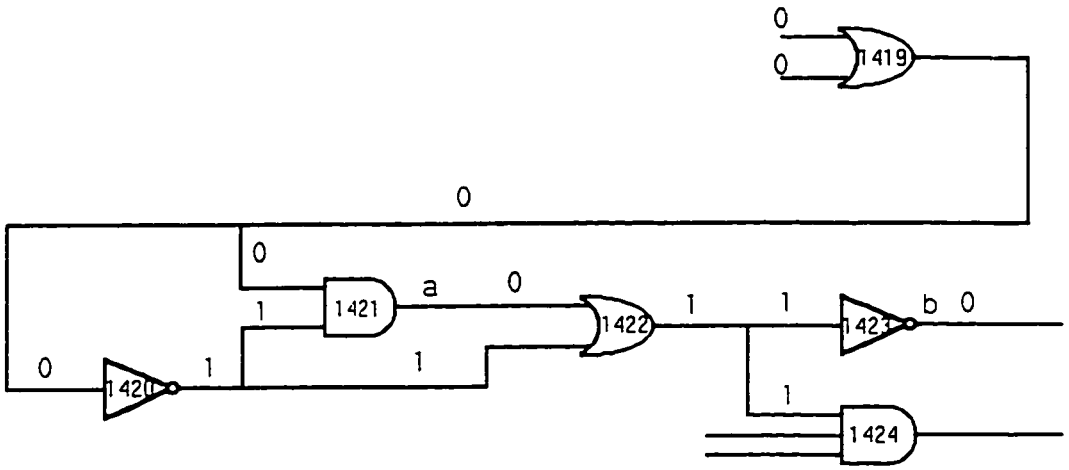


Fig. 3.4.4(c)

The final result of Rule-A/3.4 is shown in the circuit of Fig. 3.4.4(d) (The faults with * are not shown in [19]).

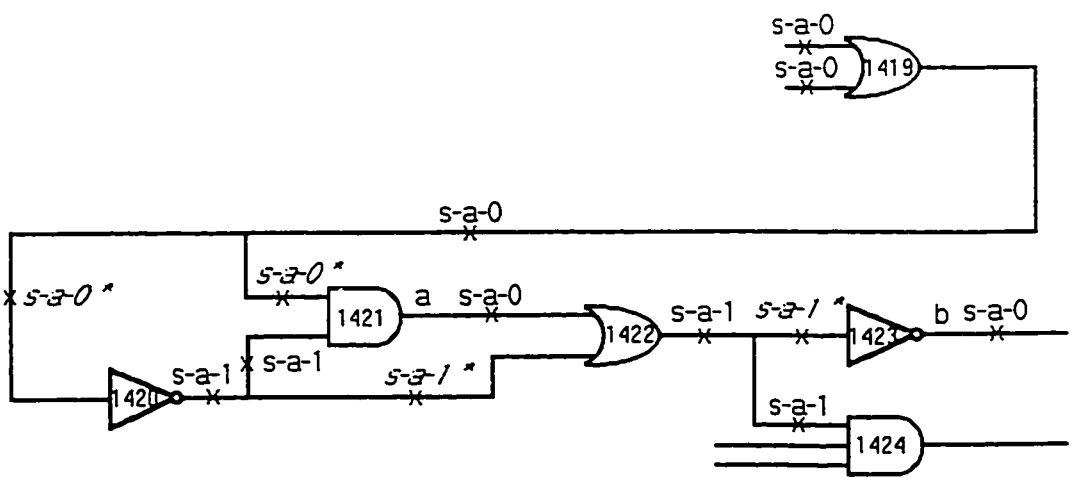


Fig. 3.4.4(d)

3.5 DYRID (DYnamic Redundancy IDentification) [1]

DYRID is a procedure which concentrates on the improvement of the RI process. Its basis is outlined in the discussion provided below.

Suppose T_β denotes the set of test patterns for fault β and suppose T_α denotes the set of test patterns for fault α. If $T_\alpha \subseteq T_\beta$ then the fault α is said to **test-cover** the fault β.

Suppose it is known that fault α test covers fault β. If it is then established that fault α is redundant; i.e., $T_\alpha = \phi$ (empty), then it will immediately follow that $T_\beta = \phi$ (because $T_\alpha \subseteq T_\beta$); i.e., fault β is also redundant. We refer to this observation as **DY1/3.5.**

Fig. 3.5.1 provides an example.

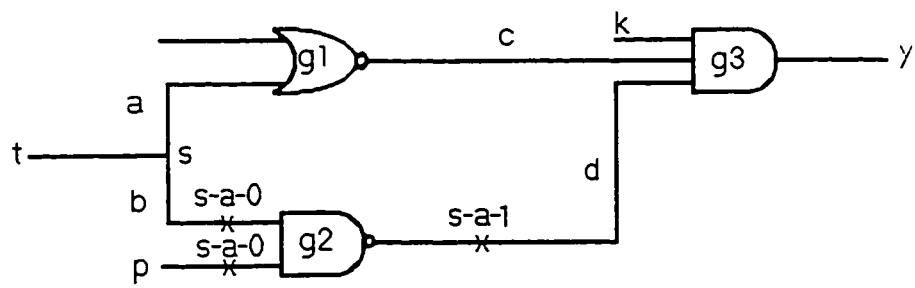


Fig. 3.5.1

The requirement for fault sensitization is b = 1, and the requirement for fault propagation of (b: s-a-0) is pkc = 111. The requirement for fault sensitization is p = 1, and the requirement for fault propagation of (p: s-a-0) is bkc = 111. In other words, both require that bpkc = 1111. This, in turn, implies that both (b: s-a-0) and (p: s-a-0) require the same test patterns (Multiple test patterns may be possible because bpkc = 1111 may imply several different assignments at the primary input lines of the circuit, and each one can be used as a test pattern for both faults).

Now look at (d: s-a-1), The requirement for fault sensitization is d = 0; i.e., b = 1 and p = 1. The requirement for fault propagation is kc = 11. Therefore, the test patterns for (b: s-a-0) and (p: s-a-0) are also test patterns for (d: s-a-1). In other words, (d: s-a-1) test-covers (b: s-a-0) and (p: s-a-0). According to DY1/3.5, after the fault (d: s-a-1) is identified redundant, (b: s-a-0) and (p: s-a-0) can also be marked redundant.

In [1], the author provides an additional useful strategy for handling faults which may occur at a stem. This is contained in the following observation taken from [1].

Let Si be a FOB (Fan Out Branch) of stem s. If for every primary output line fed by Si, Si is either nonreconvergent or reconverges only with equal inversion parities*, then (s: s-a-v) test-covers (Si: s-a-v) and Si is called **safe FOB**. So if a stem fault is redundant, then the corresponding faults on all its safe FOB's are also redundant. We refer to this observation as **DY2/3.5.**

Fig. 3.5.2(a) is an example circuit presented in [1]. After (a: s-a-1) has been proven redundant, (c: s-a-0), (s: s-a-0) can also be marked redundant (This is an application of Rule-A/3.4 which is incorporated in the DYRID procedure). Then according to DY2/3.5, (s1: s-a-0), (s2: s-a-0 are identified redundant. Similarly (t: s-a-1), (t1: s-a-1), (t2: s-a-1) and (q: s-a-1) redundant. Here s1 and s2 are safe FOB's of s, t1 and t2 are safe FOB's of t because they are nonreconvergent. To test fault (u: s-a-1) it requires u = 0 and t2 = 0, which is also the requirement for testing (q: s-a-1), so (q: s-a-1) test-covers (u: s-a-1). Applying DY1, (u: s-a-1) is identified redundant because (q: s-a-1) is redundant. Applying DY1 again, (r: s-a-0), (p: s-a-1), (o: s-a-1), (e: s-a-0) and (f: s-a-0) are all redundant faults because (u: s-a-1) test-covers these faults (This can be demonstrated in a similar way to the demonstration that (q: s-a-1) test-covers (u: s-a-1)).

The circuit which results from there operations is shown in Fig. 3.5.2(b).

---

* Inversion parity is 0 if the number of NOT, NAND, and NOR gates on the path in question is even, otherwise is 1.

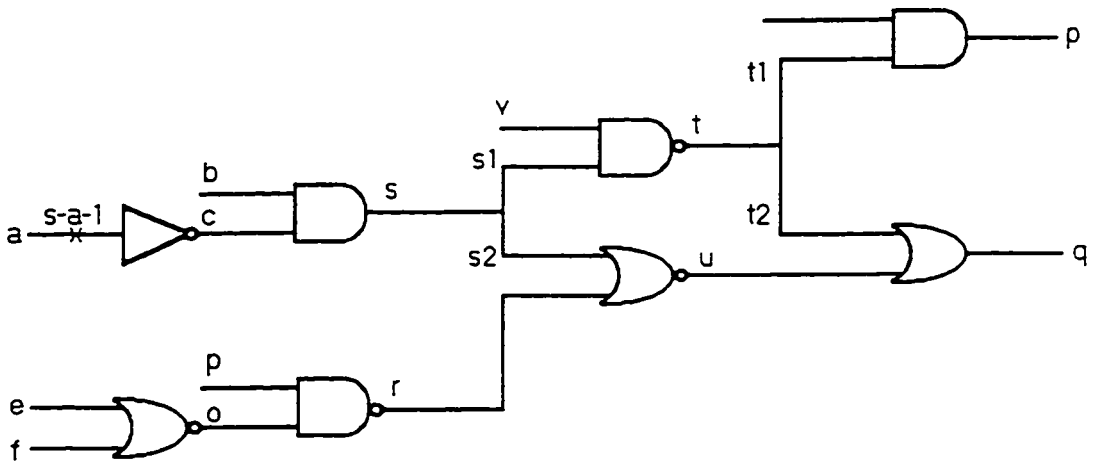


Fig. 3.5.2(a)

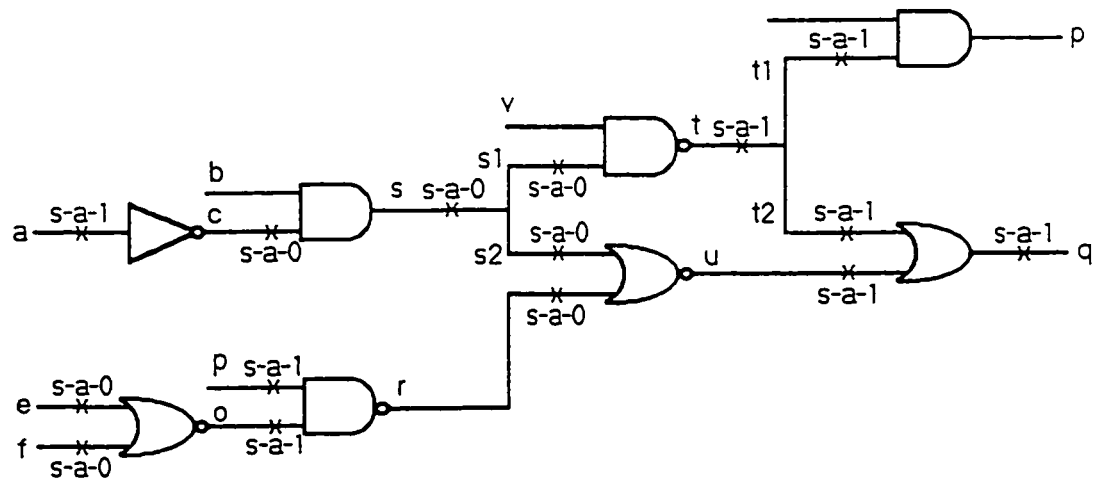


Fig. 3.5.2(b)

3.6 Harihara and Menon's Structure-Based Method [23]

In [23], Harihara and Menon proposed a novel approach to the RI problem based on an examination of circuit structure. Our main contribution, as presented in chapter 5, extends the underlying ideas of the structure-based approach as presented by Harihara and Menon (we refer to the later as the HM method).

The HM method is introduced in the following three subsections entitled Definitions, Methods and Examples. The contents of all three subsections have been taken exactly from [23]. The following, however, should be noted:

1. The term "undetectable faults" has exactly the same meaning as "redundant faults" as used in this thesis.

2. The definitions presented in subsection **3.6.1** (**Definitions**) are applicable only in section **3.6 Harihara and Menon's Structure-Based Method**. They are either not used in the rest of this thesis (e.g., sensitive, essential value, etc.) or defined in different way (e.g., reconvergent region (G, s).

3. The labels marked in bold font in front of each step of the methods as presented in subsection **3.6.2** (**Methods**) are added for the convenience of comparison between the HM method and our method which is presented in chapter 5.

3.6.1 Definitions

Definition 1: A lead with two or more fanout branches (FOBs) is called **a fanout stem** or simply **a stem**.

Definition 2: A gate is called a **reconvergent gate** of a stem s, if there exist two or more disjoint paths from s to G.

There may be more than one reconvergent gate associated with a stem. A reconvergent gate may have multiple stems.

Definition 3: The set of inputs of a reconvergent gate G, reachable from its stem s is called **the reconvergent input set of (G, s)**, denoted by I(G, s).

Definition 4: The **even parity input set, I_0(G, s)** is the largest subsets of I(G, s), such that for every gate input i $\in$ I_0(G, s) there is a path from s to i with even inversion parity. The **odd parity input set, I_1(G, s)** is similarly defined.

The above definitions are from [35].

Definition 5: A lead j is said to be **bound** by a gate A, if all paths from j to primary outputs pass through A.

Definition 6: An input i of a gate A is said to be **sensitive** if a change in the signal value at i will result in a change in the gate output.

Definition 7: Let c be the controlling value for a reconvergent gate, G(0 for AND and NAND gates, and 1 for OR and NOR gates), and let s be a stem of G. If the value v_s at the stem s implies $v_i = c$ at an input i of G, any path from s to i, obtained by tracing back from i along sensitive inputs until s is reached is called a **controlling value path** or **cv path** for the stem value, v_s. Paths from s to G without the above property will be called **non-cv paths** for the stem value v_s.

Definition 8: If $v_j = a$ is a necessary condition for producing $v_k = b$, $a,b \in \{0, 1\}$, then $v_j = a$ is said to be an **essential value** for $v_k = b$. Let A be a gate that is reachable from a line j, and let I_j be the set of inputs of the gate A that are reachable from j. The value v_j is said to be an **essential sensitizing value** for A, if j must be assigned the value v_j to propagate a signal change on any gate input $i \notin I_j$ through A.

3.6.2 Methods

BM0 As the first step of our analysis, we identify fanout stems with reconvergent branches. For each pair (G, s), where G is a reconvergent gate of the stem s, we determine the even and odd inversion parity input sets, $I_0(G, s)$ and $I_1(G, s)$.

Single-Stem Analysis

BM1 Procedure 1: Cv and Non-cv Paths

Let G be a reconvergent gate of a stem s, and let c be the controlling value up to G.

BM1a For p = 0, 1

If $Ip(G, s) \neq \Phi$, set $vs = c \oplus p$ and perform forward implication up to G.

BM1b Trace back from each input of G with value c (if any), along paths defined by sensitive gate inputs, until the stem is reached, to identify cv paths. All other paths are non-cv paths.

For v_j to be an essential sensitizing value for A, setting j to v_j must be essential value for $v_k = \bar{c}$, for all $k \in I_j$. The following procedure determines values on lines and sensitization conditions for gates for which a given line has an essential value.

BM2 Procedure 2: Essential Values

BM21 1. Set the lead j to $v_j = a$, $a \in \{0, 1\}$, and perform implication.

BM22 2. For every lead k such that $v_k = b$, $b \in \{0, 1\}$, $v_j = \bar{a}$ is an essential value for $v_k = \bar{b}$.

BM23 3. For every gate A with one or more input values equal to the controlling value c, $v_j = \bar{a}$ is an essential sensitizing value.

BM3 Procedure 3: Single-Stem Analysis

Let G be a reconvergent gate of a stem s.

BM31 1. Determine cv and non-cv paths from s to G, using procedure 1.

BM31a If there are no cv paths

BM31b No undetectable faults associated with this region are identified by single-stem analysis; END.

BM32 2. For every stem value v_s, that produces a cv path

BM32a Determine N_f, the number of FOBs of s that lie on cv paths.

BM32b Using Procedure 2, determine the lines j and their values v_j for which v_s is essential. Also, determine every gate A for which v_s is an essential sensitizing value.

BM32c For every gate A for which v_s is an essential sensitizing value , both stuck-at faults on every lead i bound by A, but not reachable from s are undetectable.

BM32d If $N_f = 1$, stuck-at-$\bar{v}_j$ on any line j for which v_s is essential for setting j to v_j is undetectable, if it is not on the cv path and is bound by G.

BM32e If $N_j > 1$, stuck-at-$\bar{v}_j$ on any line j, bound by G and for which v_s is essential for setting j to v_j, is undetectable.

BM33 3. If the controlling value is implied at one or more inputs of G by both $v_s = 0$ and $v_s = 1$

BM33a If s is bound by G, both stuck-at faults on the stem s are undetectable. Faults on any lead j, such that all paths from j to any primary output pass through s are also undetectable.

BM33b Let v_G be the value at the output of G when any of its input is c. The output of G s-a-v_G is undetectable. If $\bar{v}_G$ is an essential sensitizing value for any gate A, then both stuck-at faults on any lead i bound by A and not reachable from G are undetectable.

BM34 4. For every fault that has been identified as undetectable, find all faults that are equivalent to it, and mark them as undetectable.

Multiple-Stem Analysis

If a reconvergent gate has multiple stems, producing specific values on some lines or propagating fault effects through some gates in the reconvergent region may require specific values on more than one fanout stem. This combination of values may produce the controlling value at some input of the reconvergent gate, blocking fault effect propagation through it. Many of these phenomena can be identified by multiple-stem analysis.

BM4 Procedure 4: Multiple-Stem Analysis

BM41 1. For every stem si of the reconvergent gate G

Determine line values for which the stem value is essential and gates for which the stem value is an essential sensitizing value, using Procedure 2. This step produces combinations of stem values necessary for generating certain internal signal values and/or propagating fault effects through certain gates.

BM42 2. For each line j, with value v_j for which a combination of stem values is essential

BM42a Set the stems to the essential stem value combination, If a conflict occurs, s-a-$\bar{v}_j$ is undetectable, if it is bound by G.

BM42b Perform forward implication.

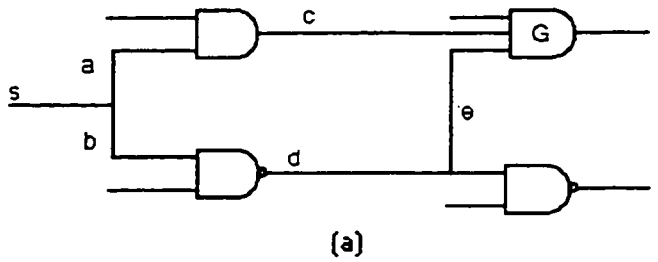
BM42c If there is at least one input of G, not reachable from j, has the controlling value for the gate G, then the fault j s-a-$\bar{v}_j$ is undetectable, if j is bound by G.

BM43 3. For every gate A with a combination of essential sensitizing values on stems

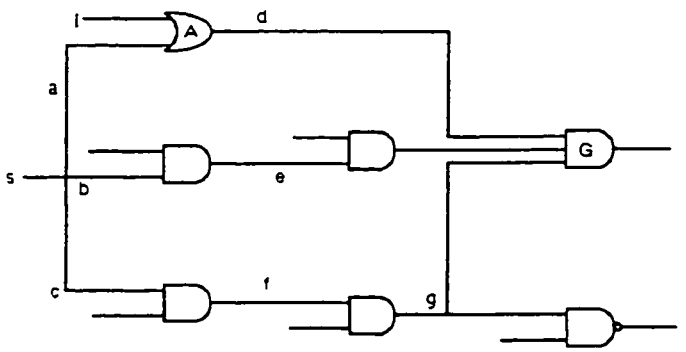
BM43a Set the stems to the essential stem value combination, and perform forward implication.

BM43b If one or more inputs of G have the controlling value, both stuck-at faults on every line k, not reachable from stems whose values were set and bound by A are undetectable.

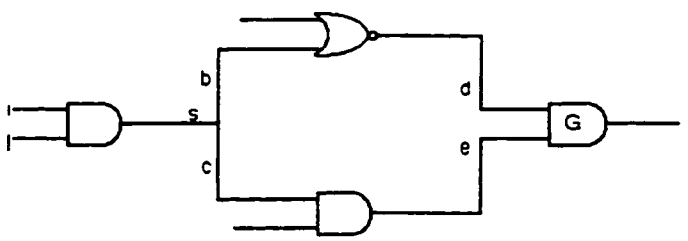
3.6.3 Examples



(a)



(b)



(c)

Figure 1

In Figure 1(a), $I_0(G, s) \neq \Phi$ and $I_1(G, s) \neq \Phi$. Therefore, both 0 and 1 must be applied to the stem s. s = 0 implies c = 0, which is the controlling logic value for the AND gate G. $N_f = 1$, and b s-a-1 is undetectable at the output of G, but cannot be marked as undetectable because it is not bound by G.

In Figure 1(b), $I_1(G, s) = \Phi$, but $I_0(G, s) \neq \Phi$. Setting s = 0, we get a = b = c = e = f = g = 0.

Two cv paths through b and c are produced, and therefore $N_f = 2$. Since the FOB c is not bound by G, we identify a s-a-1 and b s-a-1 as undetectable. s = 1 implies a = b = c = d = 1, indicating that s = 0 is essential for setting a = 0, b = 0, c = 0 and d = 0, and is also an essential sensitizing value for gate A. Therefore, both stuck-at faults on j are undetectable.

Figure 1(c) has even and odd inversion parity paths from s to G. One cv path is produced by $v_s = 0$, and $N_f = 1$, indicating that b s-a-1 is undetectable. Setting $v_s = 1$ produces a single cv path through the FOB b. Therefore, c s-a-0 is undetectable. Since the controlling input value is produced at G in both cases, both stuck-at faults on i, j and s are undetectable. The output of gate G is always 0, and s-a-0 on the gate output is undetectable.

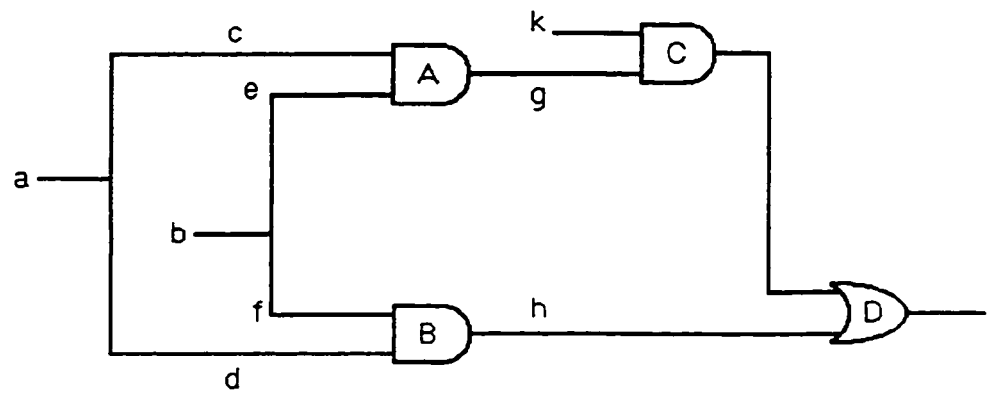


Figure 2

In the circuit of Figure 2, the reconvergent gate D has two stems a and b.

There are no cv paths from either stem to the reconvergent gate, D. Using Procedure 2 we determine that a = 1 is essential for setting c = 1, d = 1, g = 1, and i = 1. Since c = 1 and g = 1 are non-controlling inputs for gates A and C respectively, a = 1 is an essential sensitizing value for gate A and C. Similarly, b = 1 is essential for setting e = 1, f = 1 and h = 1, and also an essential sensitizing value for gate B. To detect e s-a-0, we must set b = 1 to active the fault, and a = 1 to propagate it through A. But a = b = 1 implies h = 1, and the fault effect cannot propagate through D. Hence, e s-a-0 is undetectable.

CHAPTER IV
EQUIVALENCE LAWS

In this chapter, we develop some equivalence laws for equivalent redundant fault identification. These laws provided an important basis for our method which is introduced in chapter 5.

4.1 Forward Equivalence Law I and II

In the EST approach, the following rule is introduced (Rule-A/3.4):

Suppose L1 and L2 are two lines in a circuit with the property that a value i on line L1 implies a value j on line L2. Then the existence of the redundant fault (L1: s-a-i) implies the existence of the redundant fault (L2: s-a-j).

Here we present a modified version of this rule which is restricted to a reconvergent region. It is presented as our **Forward Equivalence Law I (FEL I)**:

Consider an sG-path in the reconvergent region R(s, G) which does not have an exit path. Suppose the sG-path begins with line a and let L be either a line on the same sG-path or the output line of G. If the stuck at value i of a redundant fault (a: s-a-i) implies a value j on line L, then the occurrence of (L: s-a-j) is an equivalent redundant fault of (a: s-a-i).

FEL I is a special case of Rule-A/3.4 noted above. Consequently its correctness follows from the proof of the rule which is given in [19].

The case where an sG-path has an exit path needs special attention. For this purpose, consider a general configuration shown in Fig. 4.1.1(a)* where P(p, q) is an sG-path in a reconvergent region R(s, G).

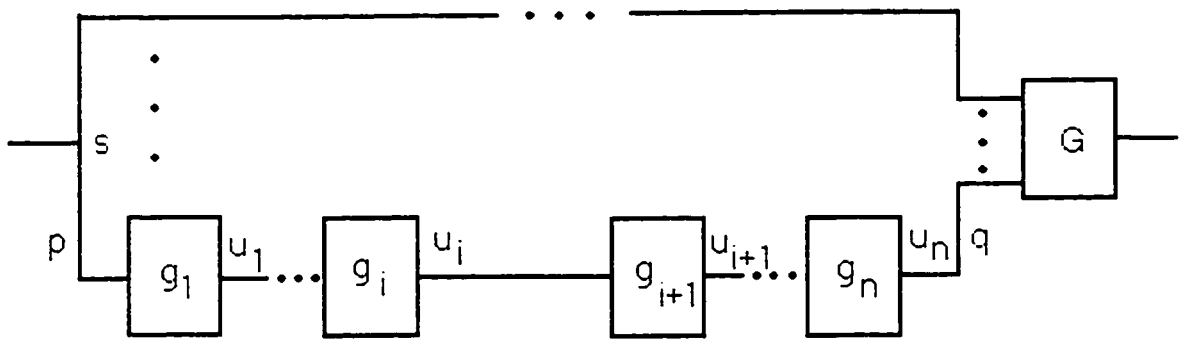


Fig. 4.1.1[a]

If a value u on line p implies the value u_i at the output of gate g_i for i = 1, 2, ..., n, then a redundant fault (p: s-a-u) implies (from FEL I) redundant faults (s-a-u_i) at the output line of g_i for i = 1, 2, ..., n. This means that it is not possible to both sensitize and propagate the faults (s-a-u_i).

---

* Here, and in the sequel, we use a rectangle to present an arbitrary gate.

Consider a modified situation where an exit path is introduced into the circuit of Fig. 4.1.1(a) at point s', which is on the output line of gate g_m, as shown in Fig 4.1.1(b):

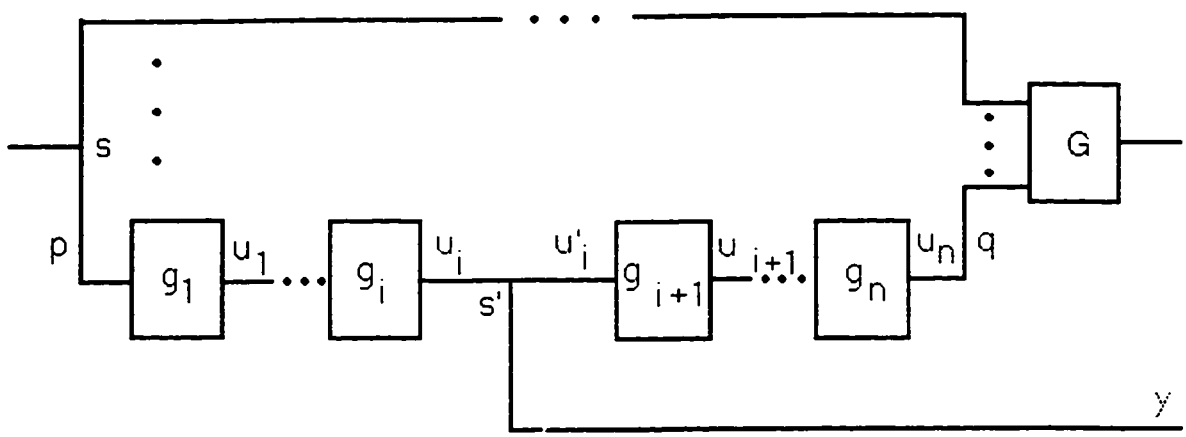


Fig. 4.1.1(b)

With this change, it is now possible to sensitize and propagate the faults (s-a-u_i) at the output of g_i for i = 1, 2, ..., m, because of the propagation path provided by the exit path. In other words, the (s-a-u_i) faults at the output of g_i, i = 1, 2, ..., m, are no longer redundant. But, on the other hand, the (s-a-u_i) faults at the outputs of g_i, i = m + 1, ..., n (and the fault s-a-u_m on the OP input line of g_{m+1}, which is marked as u'_m in Fig. 4.1.1(b)) are unaffected and remain redundant.

This provides the basis for another Forward Equivalence Law:

Forward Equivalence Law II (FEL II):

Consider an sG-path in the reconvergent region R(s, G) which has an exit path at point s'. Assume the sG-path begins with line a and let line L be a line on the same sG-path or the output line of G. Suppose that the removal of the exit path at s' allows the conclusion from FEL I that (L: s-a-j) is an equivalent redundant fault of the redundancy (a: s-a-i). If Line L is not between s and s' then the existence of the exit path does alter the conclusion that (L: s-a-j) is redundant. On the other hand, if line L is between s and s', then (L: s-a-j) is not redundant.

We illustrate FEL II with Fig. 4.1.2(a):

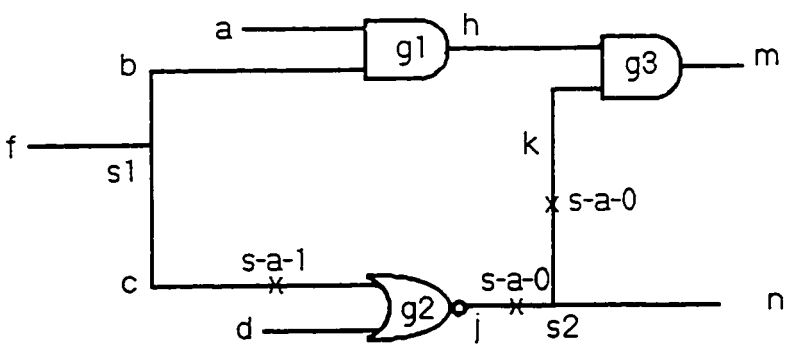


Fig. 4.1.2(a)

In the 4.1.2(a), both P(b, h) and P(c, k) are sG-paths. Furthermore, P(b, h) is a cv path and its path-cv = 0. An exit path (i.e., line n) branches out at point s2 on the sG-path P(c, k). Assume that the exit path n which branches out at the point s2 is removed. Observe that value 1 at line c implies values 0 at both line j and line k. Consequently from FEL I, the redundant fault (c: s-a-1) implies the redundant faults (j: s-a-0) and (k: s-a-0), as shown in Fig.4.1.2(a).

The application of FEL II yields the conclusion that (k: s-a-0) is redundant as shown in Fig. 4.1.2(b).

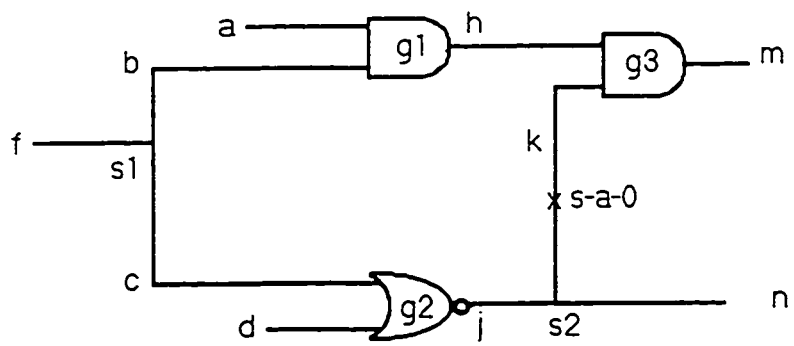


Fig. 4.1.2(b)

4.2 Backward Equivalence Law I

In Fig. 4.2.1(a), A is a gate with its input lines and its output line L. Assume that (L: s-a-β) has been identified redundant, then any redundant fault on any input line of gate A can not be propagated through gate A. This fact reflects the following truth, which we called **Backward Equivalence Law I**:

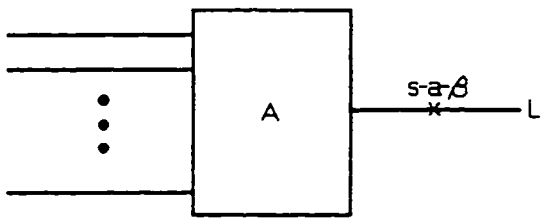


Fig. 4.2.1(a)

Backward Equivalence Law I (BEL I):

Let A be a gate with output line L. If (L: s-a-β) has been identified as redundant, then s-a-β at every input line of A is an equivalent redundant fault of (L: s-a-β) (see Fig. 4.2.1(b)):

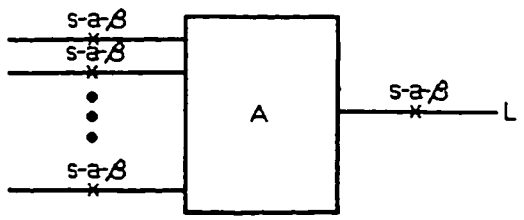


Fig. 4.2.1(b)

4.3 Backward Equivalence Law II (BEL II)

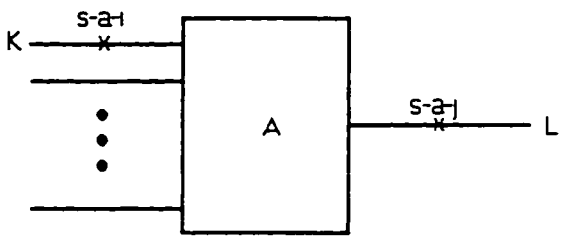


Fig. 4.3.1(a)

In Fig. 4.3.1(a), A is a gate on an sG-path in some reconvergent region R(s, G). Also assume that:

- the input line K is an OP input line to A.
- all other input lines are FP input lines to A.
- (K: s-a-i) has been identified redundant.
- i is the controlling value of A and produces an output value j.

According to FEL I, (L: s-a-j) is redundant (because value i on line K implies value j on line L). A stuck-at value of j on line L means that any sensitized stuck-at fault any OF input line of A cannot be propagated through A, hence s-a-β is redundant on every OF input line of A.

We summarize the above observations as follows:

Backward Equivalence Law II (BEL II):

Consider an sG-path in a reconvergent region R(s, G) and let K be an OP input line of gate A. If (K: s-a-i) has been identified redundant by the application of FEL I and i is the controlling value of A, then all CF input lines of A are s-a-β equivalent redundant to (K: s-a-i) (Shown as Fig. 4.3.1(b)):

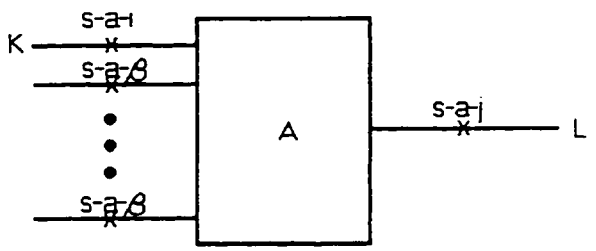


Fig. 4.3.1(b)

Fig. 4.3.2(a) - (d) illustrate the use of FEL I, BEL I and BEL II.

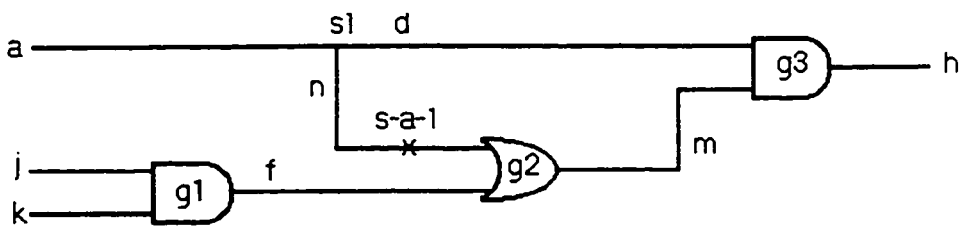


Fig. 4.3.2(a)

Assume in Fig. 4.3.2 (a), that (n: s-a-1) has been found redundant. Because line n is the beginning line of the sG-path P(n, m) for the reconvergent region R(sl, g3), the FEL I is applicable and results in Fig. 4.3.2(b).

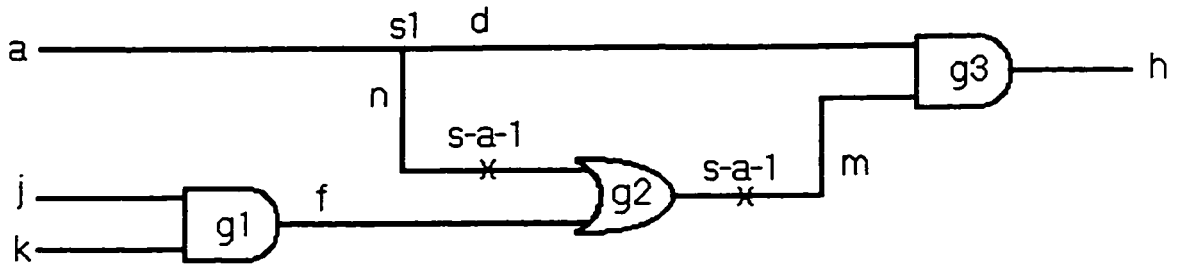


Fig. 4.3.2 (b)

Apply BEL II, the application results in Fig. 4.3.2(c).

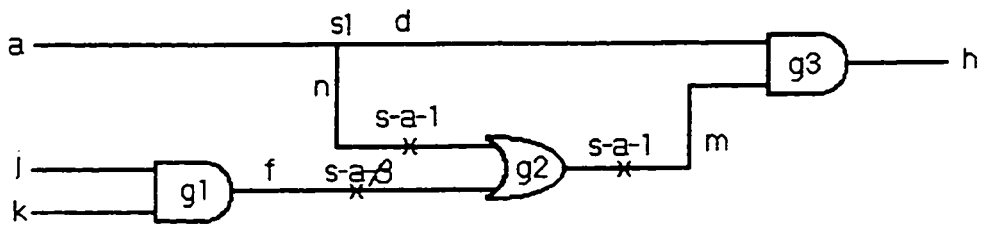


Fig. 4.3.2 (c)

Now apply BEL I, which results in Fig. 4.3.2(d).

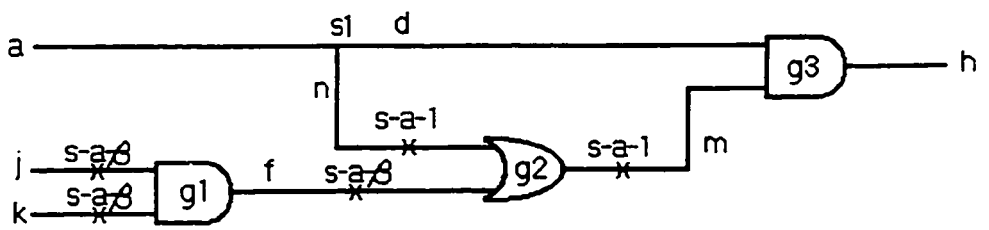


Fig. 4.3.2 (d)

4.4 Forward Equivalence Law III

Consider Fig. 4.4.1, shown as below:

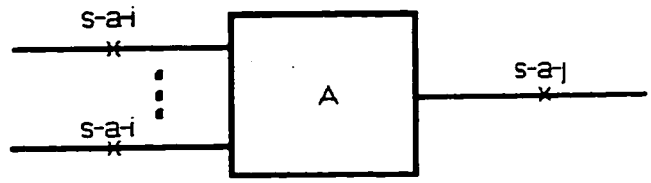


Fig. 4.4.1

According to Rule-A/3.4, the occurrence of s-a-j on the output line of A must be redundant because all the input lines of A are bound by the same reconvergent gate G.

We summarize this observation as the **Forward Equivalence Law III (FEL III)** as follows:

If A is a gate with s-a-i redundancy on each of its input line, where i is the non-controlling value of A, then there is an equivalent redundant fault s-a-j at the output line of gate A, if j is the output value for that all input lines of A are assigned the value i.

The multiple-stem region R(s1, s2, g3) in Fig. 4.4.2 is used to illustrate FEL III.

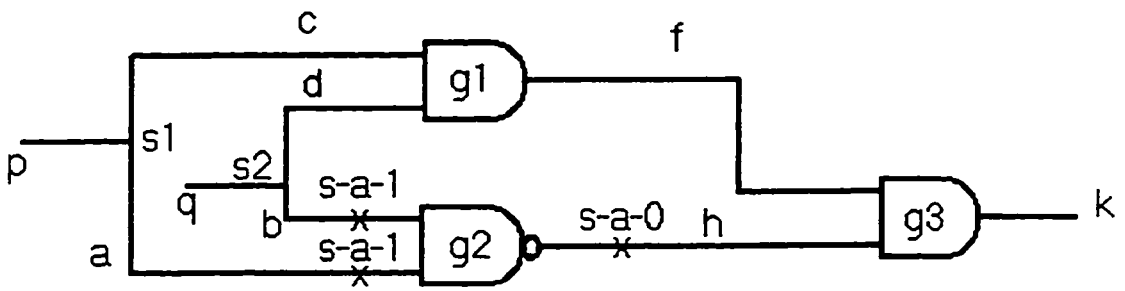


Fig. 4.4.2

If (a: s-a-1) and (b: s-a-1) have been found redundant, then FEL II can be applied because all input lines of g2 have stuck-at non-controlling value redundancies. Consequently, (h: s-a-0) is identified redundant.

4.5 Backward Equivalence Law III

To develop our Backward Equivalence Law III, consider the reconvergent region R(s, G) in Fig. 4.5.1(a). Assume that the stemline, L, of s is bound by G and that (P: s-a-j) has been found to be redundant.

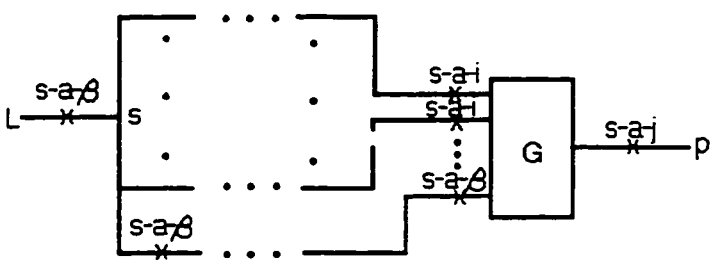


Fig. 4.5.1(a)

Because L is bound by G, any stuck-at fault on L cannot be propagated to a primary output through G; hence is redundant. In other words, the redundant fault (p: s-a-j) implies the redundant fault (L: s-a-β).

The s-a-j redundant fault at the output of G is a result of applying FEL I or FEL II. There must be at least one cv path be active when L = 0 (here a cv path is **active** means that an assignment at the stemline will uniquely determine the value of the output of G along that cv path), and at least another cv path be active when L = 1; i.e., no matter the value at the stemline is 0 or 1, a cv path must be active to imply the s-a-j redundancy at the output of G. Consequently, all single stuck-at faults, which are on all sG-paths different from these two cv paths, can not be propagated through G. This results in s-a-β redundancy at all the OP input lines on the other sG-paths as shown in Fig. 4.5.1(a).

In addition, consider that case that more than one cv paths will be active with an assignment L = u and at least one other cv path will be active with the assignment L = $\bar{u}$. For example, cv path P1 and cv path P2 both will be active when L = u, and cv path P3 will be active when L = $\bar{u}$. Then any single stuck-at fault on P1 cannot be propagated through G because the sensitizing value, v, for the single stuck-at fault will make either the cv path P2 (v = u) or the cv path P3 (v = $\bar{u}$) to be an active cv path which uniquely determines the value at the output line of G. This concludes that, in the case shown in Fig. 4.5.1(a), if there exist more than one cv paths with the same path-cv, the s-a-β is redundant at all OP input lines on these cv paths.

We use the sample circuit in Fig. 4.5.1(b) and Fig. 4.5.1(c) to illustrate the above observation:

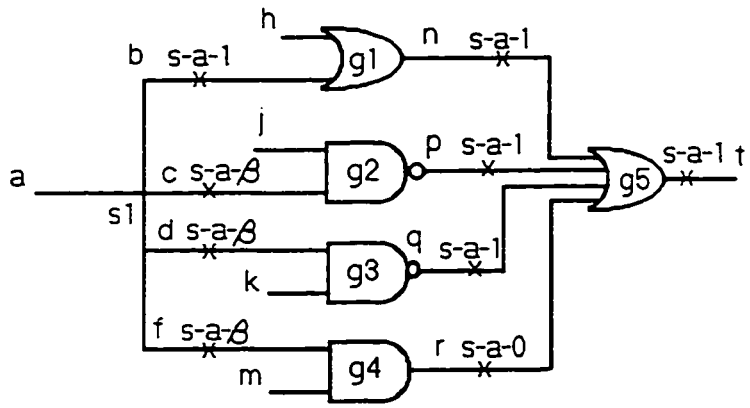


Fig. 4.5.1(b)

There are three cv paths in Fig. 4.5.1(b), which are P(b, n), which will be active when a = 1, P(c, p) and P(d, q) which will be active when a = 0. After applying FEL I, a s-a-1 redundant fault is identified at the output of g5, as shown in Fig. 4.5.1(b).

According to the previous observation with Fig. 4.5.1(a), (a: s-a-β) is redundant. Furthermore, because there exist more than one cv paths which will be active when a = 0; namely, P(c, p) and P(d, q), all OP lines on P(c, p) and P(d, q) are s-a-β redundant. As shown in Fig. 4.5.1(c).

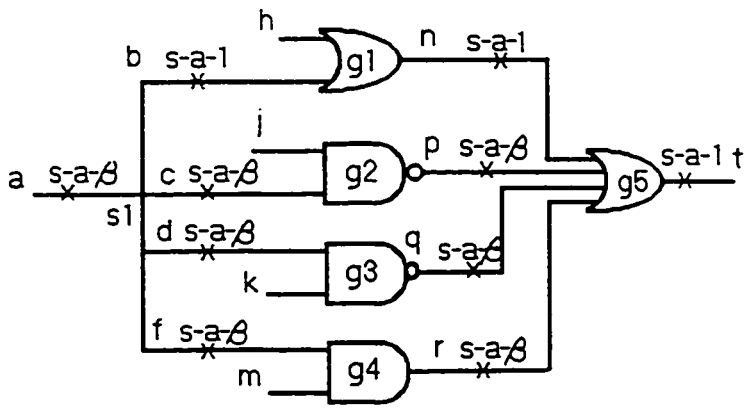


Fig. 4.5.1(c)

We summarize this observation as the **Backward Equivalence Law III (BEL III):**

Backward Equivalence Law III:

In a reconvergent region R(s, G), a redundancy at the output line of G implies a s-a-β redundant fault on the stem line of s, if s is bound by G. Furthermore, all OP input lines on any sG-path which is not a cv path are s-a-β redundant. In addition, if there exist more than one cv paths which will be active when the stemline s is assigned a value, then all OP input lines on these cv paths are s-a-β redundant.

4.6 Forward Equivalence Law IV and Backward Equivalence Law IV

In Fig. 4.6.1(a) we assume that (p: s-a-i) has been has been determined to be redundant as a consequence of the application of FEL I. Also we assume that a value i on line q implies a value k on line x and a value m on line y.

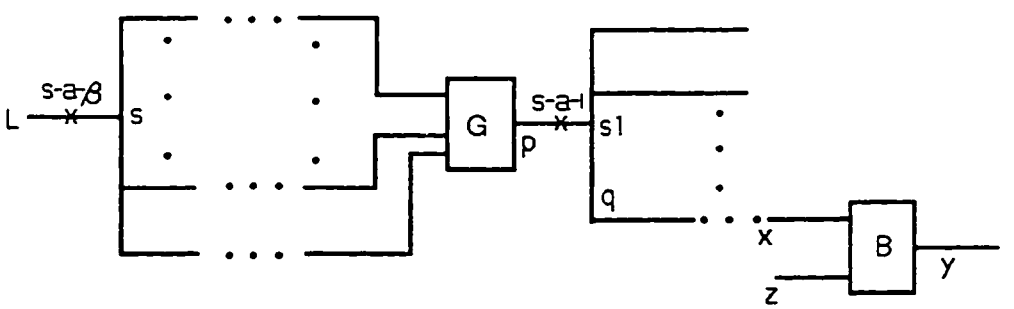


Fig. 4.6.1(a)

Observe first that a (s-a-β) redundancy arises on line L from BEL III.

With respect to stem s1, all branches of s1 have the same redundant fault s-a-i because $\bar{i}$ can never be assigned in order to any branch to sensitize the fault because of the stuck-at value (p: s-a-i). Because of our assumption that the redundant fault (p: s-a-i) was the result of the application of FEL I, Rule-A/3.4 now can be applied to find all equivalent redundancies(see Fig. 4.6.1(b)).

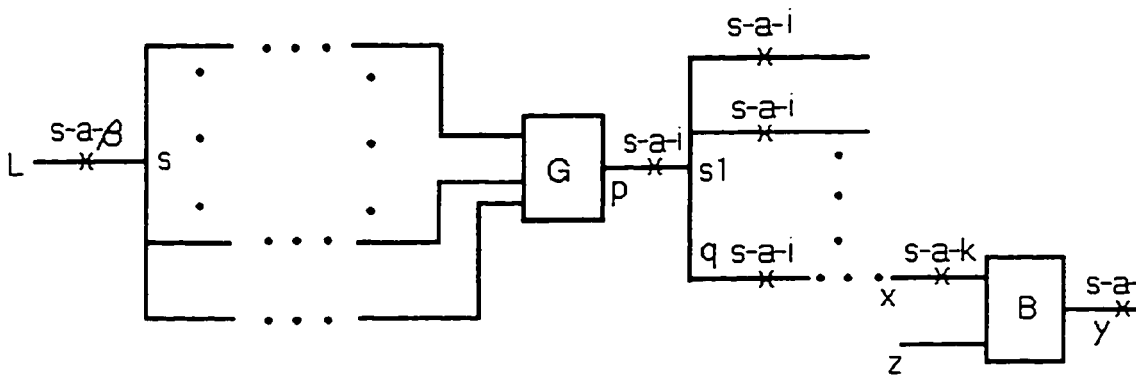


Fig. 4.6.1(b)

It must be pointed out that the (s-a-β) redundancy on the stemline, L, of s cannot be implied to the branches of s because the redundancy (L: s-a-β) is a result of taking backward equivalence from (p: s-a-i) (i.e., the application of BEL III).

Here we use the circuit in Fig. 4.6.1(c) as an example to illustrate the above point:

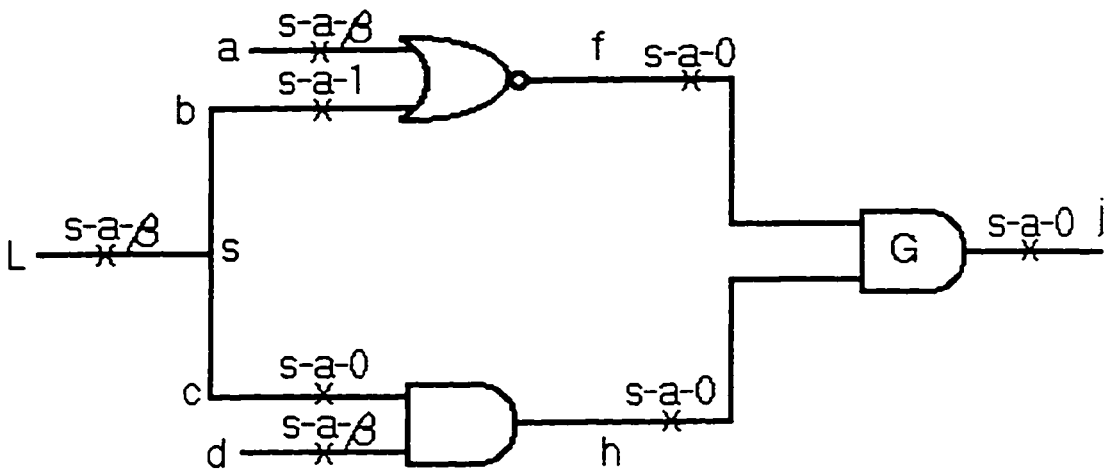


Fig. 4.6.1(c)

Assume in Fig. 4.6.1(c), that (b: s-a-1) and (c: s-a-0) have been identified as redundant. By applying FEL I, (f: s-a-0), (j: s-a-0) and (h: s-a-0) are found redundant. By applying BEL I, (a: s-a-β) and (d: s-a-β) are found redundant. Then BEL III implies (L: s-a-β). But FEL III cannot be applied to (L: s-a-β) because it is resulted by the application of BEL III. In fact, (b: s-a-0) can be tested by the test pattern (a = 0, L = 1 and d = 1), (d: s-a-1) can be tested by (a = 0, L = 0 and d = 1).

We summarize the above as **Forward Equivalence Law IV:**

Forward Equivalence Law IV (FEL IV):

Provided that a redundancy s-a-i at a stemline has been determined by the application of FEL I, then that redundancy implies the same redundancy s-a-i at each branch of the stem, and redundancy s-a-j at each line, L, if i implies j on line L.

Referring again to Fig. 4.6.1(b), the redundancy (p: s-a-i) on the output line of the reconvergent gate has been implied to each branch of stem s1 according to FEL IV. The redundancy (q: s-a-i) implies a redundant fault (x: s-a-k) on the input line of the gate B and a redundant fault (y: s-a-m on the output line of the gate B.

Note now that the propagation of any sensitized fault along any other input line of B has been blocked. Consequently, z: s-a-β) is redundant. This latter observation leads to Backward Equivalence Law IV (which is similar to BEL II):

Backward Equivalence Law IV (BEL IV):

Let t be a stemline and let L be an input to a gate A. If the Forward Equivalence Law IV (relative to stemline, t) implies (L: s-a-i) redundant and i is the controlling value of A, then (s-a-β) is an equivalent redundant fault of (L: s-a-i) for all other input lines of A which are not reachable from t.

Applying BEL IV to Fig. 4.6.1(b) results in Fig. 4.6.1(d):

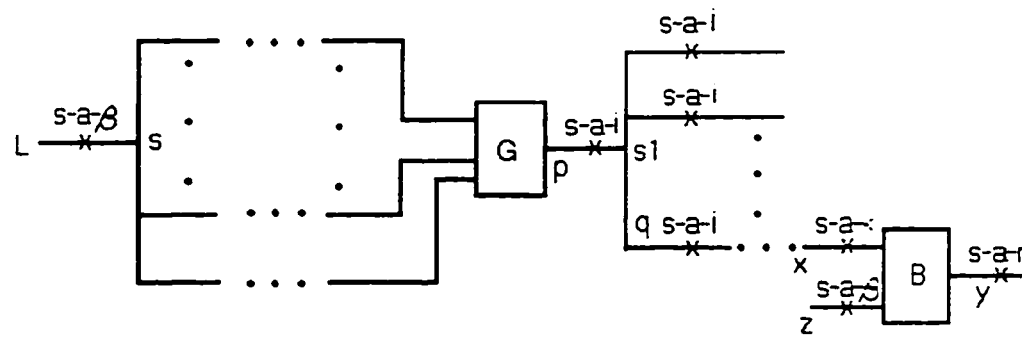


Fig. 4.6.1(d)

The relationship between the equivalence laws formulated in this chapter and similar equivalence-based method; e.g., Rule-A/3.4 in EST and DY1/3.5 and DY2/3.5 in DYRID, will be discussed in Chapter 5.

CHAPTER V
THE IMPROVED STRUCTURE-BASED METHOD

The presentation in this chapter outlines a number of improvements to the HM method introduced in chapter 3. Many aspects of our work are based on the equivalence laws developed in chapter 4. In effect, we present an RI method which, like the HM method, is based on the structure of the circuit. We call our new approach the **Improved Structure-Based** method (the **ISB** method).

5.1 Cv path identification

In the HM method, the definition of a cv path is as follows(see Definition 7 in section 3.6):

Let c be the controlling value for a reconvergent gate, G(0 for AND and NAND gates, and 1 for OR and NOR gates), and let s be a stem of G. If the value v_s at the stem s implies $v_i = c$ at an input i of G, any path from s to i, obtained by tracing back from i along sensitive inputs until s is reached is called a ***controlling value path*** *or* ***cv path*** *for the stem value, v_s. Paths from s to G without the above property will be called* ***non-cv paths*** *for the stem value v_s.*

Within the context of our ISB method, it is convenient to use an alternate (but equivalent) definition of a cv path. This is given below:

Consider a reconvergent region R(s, G) and let P(x, y) be an sG-path in R(s, G). Let g_1, g_2, ... g_n, be the sequence of gates along P(x, y). If a value, v, on line x is the controlling value of g_1 and the output of g is the controlling value for gate g_{i+1}, for i = 1, 2, ..., (n-1) and the output of g (i.e., the value on line y) is the controlling value of G, then P(x, y) is a cv path.

With respect to the above definition, we call the value v the ***path-cv*** *for the cv path P(x, y).*

With this definition, our ISB method identifies cv paths using the following procedures (simplified version, the part for cv path identification only. The original version can also identified so called initial redundancies as well, which will be introduced in the following section):

PROC. Cv-Path-Identification (R(s, G); Path-Cv-Table)*

```
START
     FOR every sG-path P(x, y)
     {
          set v to be the controlling value of g1, the FGL gate.
          let g_a be g1 and let g_b be the successor gate to g_a.
          Forward-Implication(g_a, g_b; cv-path-flag);
          IF cv-path-flag = TRUE
          {
               add P(x, y) and v into the Path-Cv-Table;
          }/end of IF/
     } /end of FOR/
END.
```

---

* Here and in the sequel, we use ";" to separate input parameters and output parameters of each procedure.

PROC. Forward-Implication (g_a, g_b; cv-path-flag)

START

```
cv-path-flag = FALSE;

WHILE the output of g_a is the controlling value of g_b
{
     IF g_b = G
     {
          cv-path-flag = TRUE;
     } /end of IF/
     Let g_a be g_b and then g_b be the successor to g_a;
} /end of WHILE/
```

END.

In the above procedure, the cv-path-flag is a variable and the Path-Cv-Table is a table with two columns. One column is the sG-path identifier to indicate which sG-paths have been found cv paths and the other column is the corresponding path-cvs.

We use the circuit in Fig. 5.1.1 to illustrate the Cv-Path-Identification-Procedure.

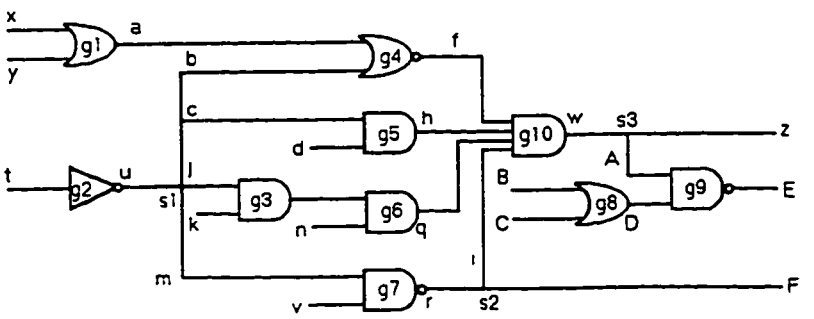


Fig. 5.1.1

The region under consideration is R(s1, g10). Consider first the sG-path P(b, f) and assign u = 1, which is the controlling value of g4. This implies f = 0. The Forward-Implication procedure returns a TRUE value for the cv-path-flag; hence P(b, f) is added to the Path-Cv-Table and its path-cv is assigned the value 1.

P(c, h) and P(j, q) are similarly added to the List-of-cv-Paths. However the procedure Forward-Implication establishes that P(m, i) is not a cv path. The updating of the Path-Cv-Table is shown as below:

After the first iteration

Path-Cv-Table

sG-path-spec.	path-cv
P(b, f)	1

After the second iteration

Path-Cv-Table

sG-path-spec.	path-cv
P(b, f)	1
P(c, h)	0

After the third iteration

Path-Cv-Table

sG-path-spec.	path-cv
P(b, f)	1
P(c, h)	0
P(j, q)	0

Table 5.1.1 Path Cv Table

Comparison with the HM method:

The main advantage of the above approach for cv path identification is that the procedure Forward-Implication travels each sG-path only once. This is in contrast to the case with the HM method where each sG-path is travelled at least twice and possibly three times (see steps BM0 and BM1 in section 3.6.2).

5.2 How Cv Paths Cause Redundancies and Initial Redundancy Identification

In a region R(s, G), assume we have an sG-path, path1, which is the only cv path. Assume its path-cv = u. (see Fig. 5.2.1(a)). The existence of this single cv path means that the output of the reconvergent gate G is uniquely determined when the stemline is assigned the path-cv value, u. Notice that u is the necessary condition to sensitize a s-a-$\bar{u}$ fault at any input line in I(g'(R))(I(g'(R)) is the set of all OP input lines which belong to the FGL gates, see section 2.2). Because of this and the assumption that the path-cv of path1 is u, it follows that all s-a-$\bar{u}$ faults at every input line in I(g'(R), except the one on path1, are redundant(see in Fig. 5.2.1(b).

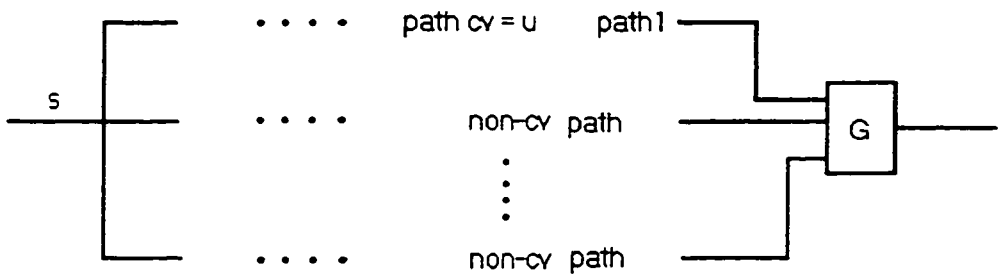


Fig. 5.2.1(a)

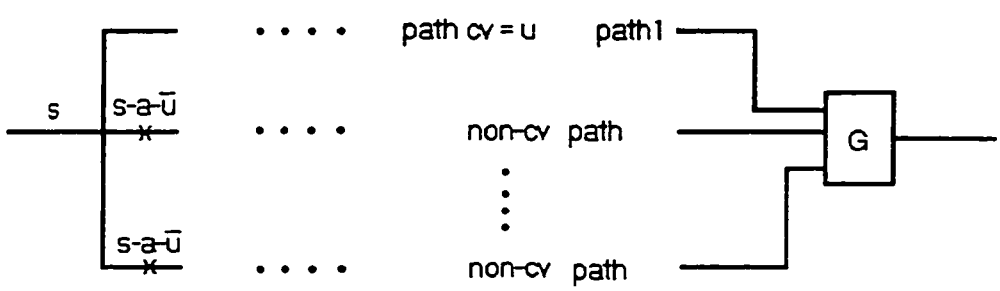


Fig. 5.2.1(b)

Suppose now that region R(s, G) contains two cv paths each with path cv = u. The beginning line of path1 is s-a-$\bar{u}$ redundant because path2 is a cv path with path-cv = u, and the beginning line of path2 is s-a-$\bar{u}$ redundant because path1 is a cv path with path-cv = u. All beginning lines on all other sG-paths continue to be s-a-$\bar{u}$ redundant. Consequently, all input lines in I(g'(R)) are s-a-$\bar{u}$ redundant(see Fig. 5.2.2).

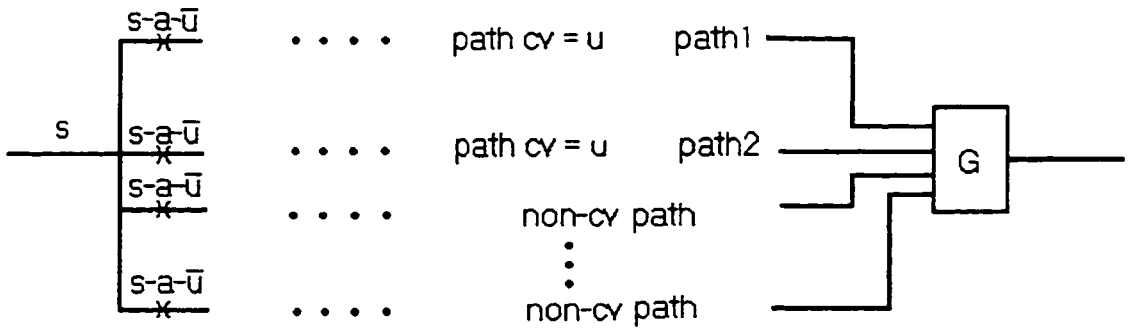


Fig. 5.2.2

Suppose now that region R(s, G) contains two cv paths with path-cv = u for path1 and path-cv = $\bar{u}$ for path2. The beginning line of path1 is s-a-u redundant because path2 is a cv path with path-cv = $\bar{u}$, and the beginning line of path2 is s-a-$\bar{u}$ redundant because path1 is a cv path with path-cv = u. All beginning lines on all other sG-paths are both s-a-u (caused by path2) and s-a-$\bar{u}$(caused by path1) redundant; i.e., s-a-β redundant. (see Fig. 5.2.3).

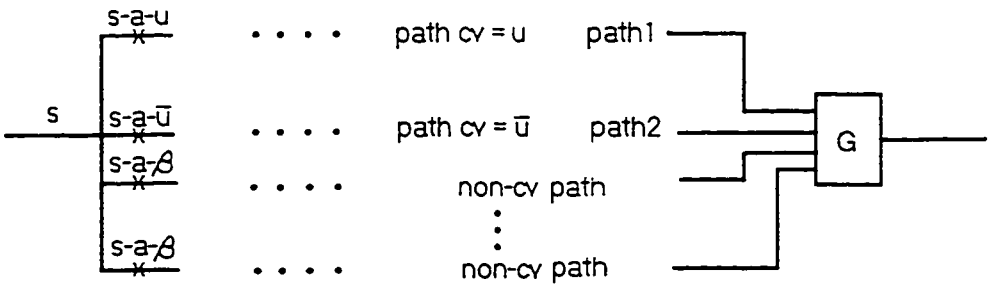


Fig. 5.2.3

We define all the redundancies on the input lines of I(g'(R)) (i.e., the beginning lines of all sG-paths) in a region R(s, G), as the **initial redundancies** of the region R(s, G) which is under consideration. For the case of Fig. 5.2.3, the initial redundancies are: s-a-u, s-a-ū, s-a-β, ..., s-a-β on the beginning lines of path1, path2, ..., etc.

Notice that the initial redundancy identification analysis can be carried out by studying the effect of each cv path independently of the other cv paths; i.e., the overall effect of cv paths path1 and path2 in the above examples can be deduced by separately considering each cv path in turn. This leads us to two important conclusions:

Conclusion-I/5.2:

A cv path with path-cv = u causes redundant fault s-a-ū at all the input lines in I(g'(R)) except the one on the cv path.

Conclusion-II/5.2:

In a reconvergent region R(s, G), all the initial redundancies at the input lines in I(g'(R)) can be identified by overlaying the effect of each cv path independently.

Our ISB method includes an initial redundancy identification procedure, which is based on the above two conclusions.

It should however, be stressed that Conclusion-I and Conclusion-II apply only if there are no exit paths(see section 2.2) on the sG-paths under consideration. If such exit paths exist, then modified conclusions must be formulated.

Consider an sG-path, P1, in a reconvergent region R(s, G) and an exit path, P2, which branches out at point t on P1. There will be no initial redundancy on P1 because any sensitized redundancies between s and t on P1 can be propagated through P2. However, in order to be able to identify possible redundancies which exist between t and G, our ISB method includes a special procedure to deal with an sG-path that contains an exit path. The basis for this procedure is the following observation:

Consider a reconvergent region R(s, G) which has a cv path, P1, whose path-cv = u. Suppose P' is another sG-path which has an exit path. In the absence of the exit line on path P', a redundancy s-a-$\bar{u}$ would exist on the beginning line of P'. However because of the exit line, this redundancy is in fact, a **virtual redundancy**. The virtual redundancy s-a-$\bar{u}$ is considered as an initial redundancy on the beginning line of P'.

As observed earlier, once a sG-path is identified as a cv path in R(s, G), we can immediately overlay its effect onto the other sG-paths. This process must however take into account the possible existence of exit paths on the sG-path. Thus the overlaying process gives two tables; namely, the Initial-Redundancy-Table and the Virtual-Initial-Redundancy-Table. The first column in these tables is an identifier which is a pointer to the beginning line of an sG-path and the second column is the specification of the initial redundancy. Correspondingly, we replace the Cv-Path-Identification procedure presented in section 5.1, with a procedure called Cv-Path-and-Initial-RI , which is given as below:

PROC. Cv-Path-and-Initial-RI (R(s, G);
Path-Cv-Table,
Exit-Path-Table,
Initial-Redundancies-Table,
Virtual-Initial-Redundancies-Table)

```
START

     FOR every sG-path P(x, y)
     {
          set v to be the controlling value of g1, the FGL gate;
          let N_a be g1 and let N_b be the successor node to N_a;
          Forward-Implication (N_a, N_b; cv-path-flag,
                               exit-path-ID);
          IF cv-path-flag = TRUE
          {
               add P(x, y) and v into the Path-Cv-Table;
          } /end of IF/
          IF exit-path-ID ≠ NIL
          {
               add P(x, y) and exit-path-ID into the Exit-Path-
               Table;
          } /end of IF/
     } /end of FOR/

     Initial-RI (R(s, G), Path-Cv-Table,
                          Exit-Path-Table;
                          Initial-Redundancy-Table,
                          Virtual-Initial-Redundancy-Table);
END.
```

```
PROC. Forward-Implication (Na, Nb; cv-path-flag, exit-path-ID)

START

     cv-path-flag = FALSE;
     exit-path-ID = NIL;

     IF Nb = STEM
          {
          let exit-path-ID = stem-ID;
          let Nb to be the successor to Nb;
          } /end of IF/

     WHILE the value on the output of Na is the controlling value
           of Nb
     {
          IF Nb = G
          {
               {
                    cv-path-flag = TRUE;
               }
          ELSE
               {
                    Let Na be Nb and then Nb be the successor to
                    Na;

                    IF Nb is a stem line point
                    {
                         let exit-path-ID = stem-ID;
                         let Nb to be the successor node to Nb;
                    } /end of IF/
               }
          } /end of IF/
     } /end of WHILE/
```

```
WHILE N_b ≠ G
{
     Let N_a be N_b and then N_b be the successor node to N_a;

     IF N_b is a stem line point
     {
          let exit-path-ID = stem-ID;
          let N_b to be the successor to N_b;
     } /end of IF/

} /end of WHILE

END.
```

Notice that the above procedure has been modified with the addition of exit path identification feature. If there are more than one exit path on a sG-path, the exit-path-ID returns from the Forward Implication procedure is the last one to be identified, i.e., the one closest to the reconvergent gate G.

```
PROC. Initial-RI (R(s, G), Path-Cv-Table, Exit-Path-Table;
                  Initial-Redundancy-Table,
                  Virtual-Initial-Redundancy-Table)

START

     For every cv path P(x, y) in the Path-Cv-Table
     {
          For every sG-path ≠ P(x, y)
          {
               IF the sG-path is in the Exit-Path-Table
               {
                    {
                    for the entry in the Virtual-Initial-
                    Redundancy-Table corresponding to the current
                    sG-path, overlay the effect of the cv-path,
                    P(x, y);
                    }
               ELSE
                    {
                    for the entry in the Initial-Redundancy-Table
                    corresponding to the current sG-path, overlay
                    the effect of the cv-path, P(x, y);
                    }
               } /end of IF/
          } /end of FOR/
     } /end of FOR/

END.
```

We use Fig. 5.2.4 (same as in Fig. 5.1.1) to illustrate the Cv-Path-and-Initial-RI procedure:

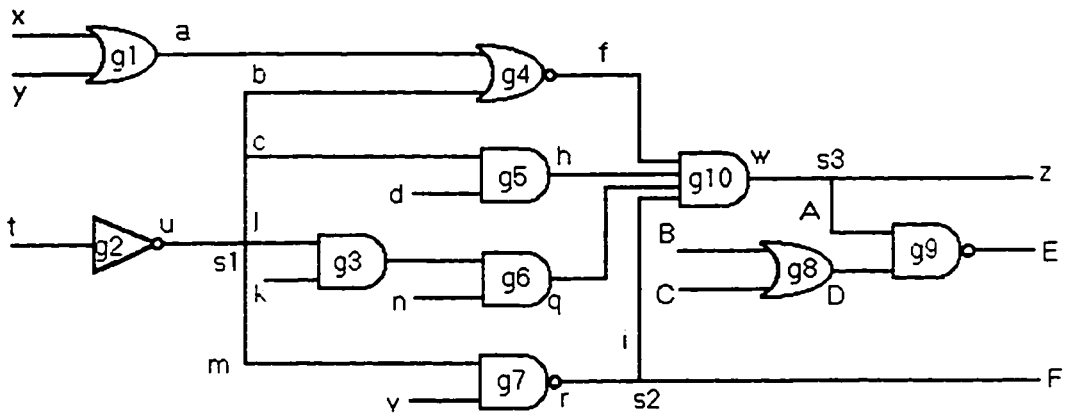


Fig. 5.2.4

Associated with each node, there is a data structure. The format of this data structure is illustrated in Fig 5.2.5 within the context of nodes g7 and s2.

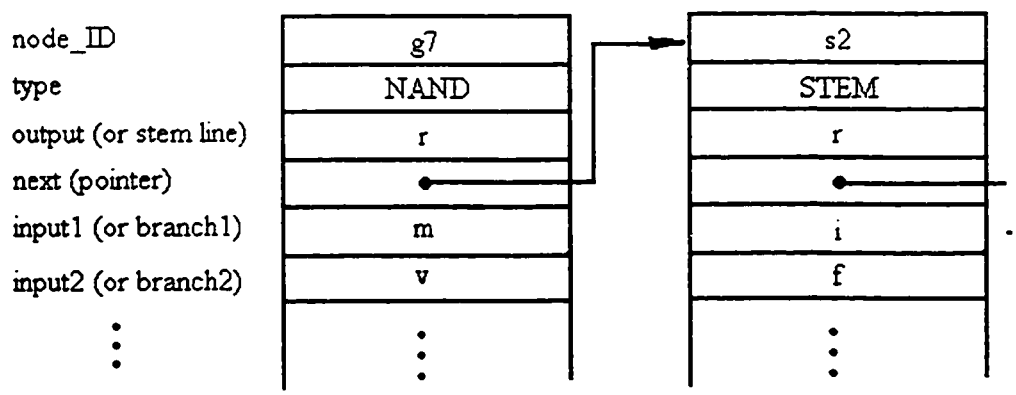


Fig. 5.2.5

The configuration of the Path-Cv-Table was introduced in section 5.1.

The Exit-Path-Table has two columns. The first column is the sG-path specification and the other is the exit-path identifier.

During the execution of the Cv-path-and-Initial-RI procedure, after the first three calls of the Forward-Implication procedure, the Path-Cv-Table is as shown earlier in section 5.1. The last call of the Forward-Implication procedure returns an exit-path-ID so we know that P(m, i) has an exit path at node s2. The resulting Exit-Path-Table is:

Exit-Path-Table

sG-path-Spec	exit-path-ID
P(m, i)	s2

Table 5.3.1
Exit Path Table

After the Path-Cv-Table and the Exit-Path-Table are generated, the Initial-RI procedure is executed as the follows:

For the cv path P(b, f):
(c: s-a-0) and (j: s-a-0) are redundant, (m: s-a-0) is virtual redundant.

For the cv path P(c, h):
(b: s-a-1) and (j: s-a-1) are redundant, (m: s-a-1) is virtual redundant.

For the cv path P(j, q):
(b: s-a-1) and (c: s-a-1) are redundant, (m: s-a-1) is virtual redundant.

Notice that the sG-path P(m, i) is not involved in the Initial-RI procedure because it is not a cv path.

During the Initial-RI procedure the Initial-Redundancy-Table and the Virtual-Initial-Redundancy-Table evolve in the following way:

	Initial-Redundancy-Table		Virtual-Initial-Redundancy-Table	
After the first iteration	lineID	Ini_Re	lineID	Ini_Re
	b		b	
	c	s-a-0	c	
	j	s-a-0	j	
	m		m	s-a-0
After the second iteration	lineID	Ini_Re	lineID	Ini_Re
	b	s-a-1	b	
	c	s-a-0	c	
	j	s-a-β	j	
	m		m	s-a-β
After the third iteration	lineID	Ini_Re	lineID	Ini_Re
	b	s-a-1	b	
	c	s-a-β	c	
	j	s-a-β	j	
	m		m	s-a-β

Table 5.3.2 Redundancy Table and Virtual Redundancy Table

From the preceding two tables, we know that (b: s-a-1), (c: s-a-β) and (j: s-a-β) are redundant and that (m: s-a-β) is virtual redundant. This is shown in Fig. 5.2.6:

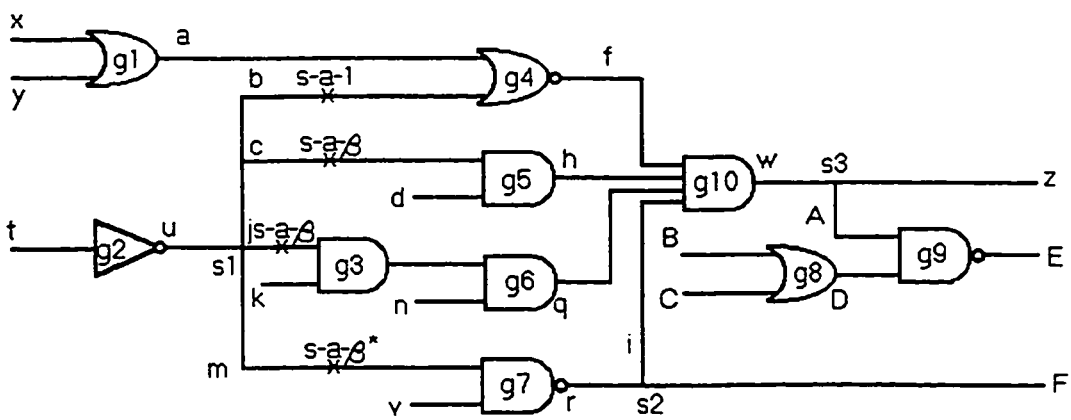


Fig. 5.2.6

Comparison with the HM method:

The advantage of our initial redundancy identification approach is that initial redundancies can be identified during the process of cv path identification. In other words, some processes in the HM method such as calculating the number of cv paths (BM32a) and sorting the processes for different strategies of RI (Bm31a, BM31b, BM32d, BM32e, BM33) have been removed.

5.3 Redundancy Identification in A Single-Stem Region

The ISB method divides the RI work into two parts: Initial redundancy identification and equivalence laws application. This strategy is based on a fact that after the initial redundancies have been found in a reconvergent region R(s, G), other redundancies in the region or resulting from the region can be further identified by applying the equivalence laws developed in chapter 4. In other words, if a redundancy (L: s-a-u) exists on a line which is inside a reconvergent region R(s, G) or bound by the region, then an initial redundancy must exist and (L: s-a-u) is an equivalent redundant fault of this initial redundancy. We provide below a brief outline of the basis for this assertion for the case where L is on an sG-path.

To test a fault s-a-u on line L of an sG-path P1 requires that a value $\bar{u}$ be established on L in order to sensitize the fault. Also the testing requires that the fault be propagated to the output of the reconvergent gate G. The propagation of the fault to the output line of G is readily achieved by assigning non-controlling values to the FP input lines of the gates between L and G.

If the fault (L: s-a-u) is redundant then it must be that it is not possible to simultaneously sensitize and propagate the fault (to the output of G). Suppose that a value $\bar{v}$ is required at the stemline of the region in order to produce the sensitizing value $\bar{u}$ at line L, i.e., $\bar{u} \Rightarrow \bar{v}$. If the fault (L: s-a-u) is redundant, this must arise because the value $\bar{v}$, via some cv-path P2, blocks the propagation of the fault through G. In other words $\bar{v}$ is the path-cv for P2. From Conclusion-I/5.2, it then follows that the beginning line of path P1 has the redundant fault s-a-v. Note now that a value v on the beginning line of P1 implies a value u on line L (this follows from the contraposition law (section 3.3) applied to $\bar{u} \Rightarrow \bar{v}$). According to FEL I, we know that (L: s-a-u) is an equivalent redundant fault of s-a-v at the beginning line of P1.

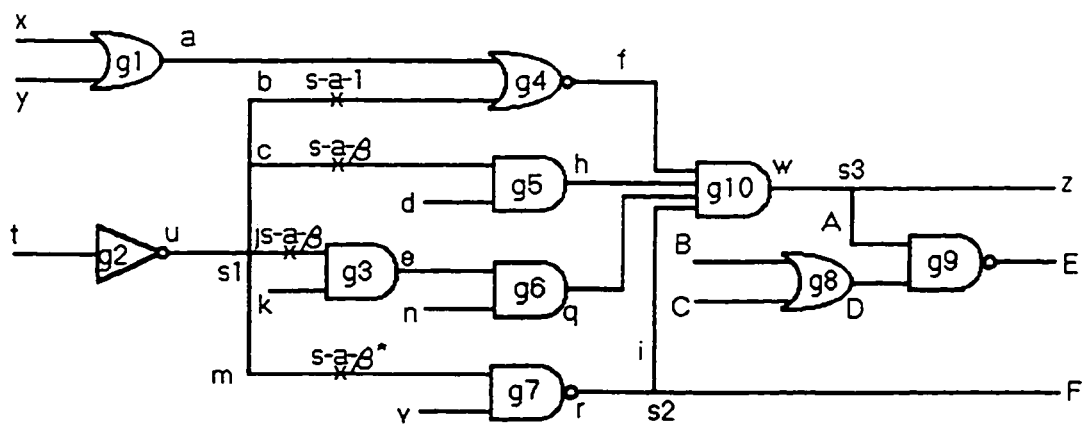


Fig. 5.3.1

In section 5.2, the procedure Cv-path-and-Initial-RI was presented as the first phase of our ISB method. We are now able to complete the specification of the method for the single-stem case by incorporating the application of equivalence laws. This is provided below:

PROC. Single-Stem-Region-RI (R(s, G); Redundancy_Table)

```
START.

    Cv-Path-Identification-and-Initial-RI (R(s, G);
                    Path-Cv-Table,
                    Exit-Path-Table,
                    Initial-Redundancy-Table,
                Virtual-Initial-Redundancy-Table);

    FOR each sG-path, P, which has an initial or virtual initial
    redundancy
    {
        IF P has no exit path
        {
            {
                Forward-Equivalence-Law-I (P,
                            Initial-Redundancy-Table;
                            OP-Redundancy-Table,
                            G-Output-Redundancy);
            }
```

```
        ELSE
             {
                  Forward-Equivalence-Law-II (P,
                                   Virtual-Initial-Redundancy-Table;
                                   Exit-Path-Table;
                                   OP-Redundancy-Table,
                                   G-Output-Redundancy);
             }
        } /end of IF/
   } /end of FOR/
   IF (G-Output-Redundancy ≠ ϕ)
   {
        Backward-Equivalence-Law-III (G-Output-Redundancy;
                                        Stemline-Redundancy,
                                      OP-Redundancy-Table)*;

        Forward-Equivalence-Law-IV (G-Output-Redundancy;
                                    Equivalent-Redundancy-Table);

        Backward-Equivalence-Law-IV(Equivalent-Redundancy-
                                   Table;
                                   NR-Redundancy-Table);
   } /end of IF/

   Backward-Equivalence-Law-II (OP-Redundancy-Table;
                                FP-Redundancy-Table);

   Backward-Equivalence-Law-I (FP-Redundancy-Table,
                               NR-Redundancy-Table;
                               X-Equivalent-Redundancy-Table);

   Report (Initial-Redundancy-Table, OP-Redundancy-Table,
           G-output-Redundancy, Stemline-Redundancy,
           Equivalent-Redundancy-Table, NR-Redundancy-Table,
           FP-Redundancy-Table, X-Redundancy-Table;
           Redundancy-Table);

END.
```

To illustrate the Single-Stem-Region procedure, we use the example introduced in section 5.1. We begin with Fig. 5.3.1, which is the result of applying the Cv-Path-and-Initial_RI procedure (see Fig. 5.2.5(b)):

The sG-paths for the region R(s1, g10) are: P(b, f), P(c, h), P(j, q) and P(m, i). The first three of these has an initial redundancy (without an exit path) and the last has an virtual initial redundancy (with an exit path). Recall that the configuration shown in Fig. 5.3.1 is the result of applying the Cv-Path-and-Initial-RI procedure. We now continue within the Single-Stem-Region procedure following the Cv-Path-and-Initial-RI procedure.

The loop over the sG-path applies the Forward-Equivalence-Law-I procedure to sG-paths P(b, f), P(c, h) and P(j, q). The Forward-Equivalence-Law-II procedure is applied to sG-path P(m, i).

The result is shown in Fig. 5.3.2 (Notice that an implicit step in applying FEL II is the removal of the virtual redundancy (m: s-a-β).

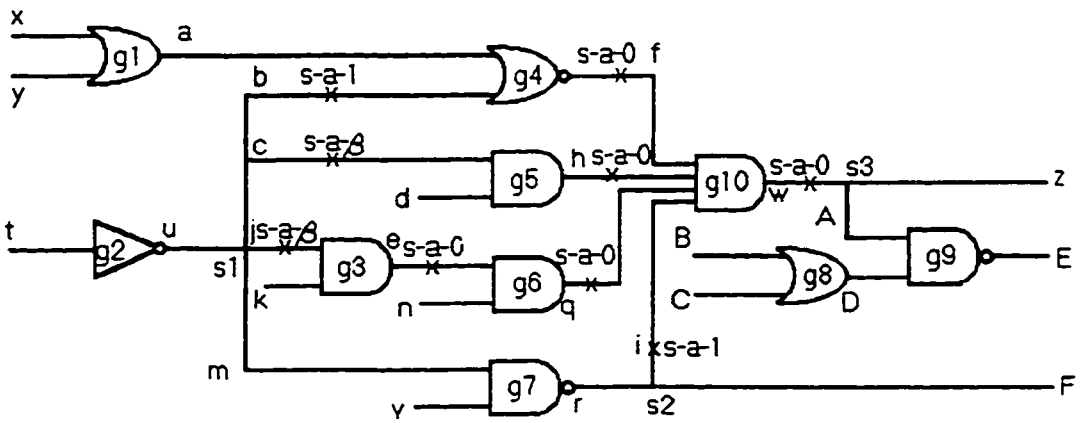


Fig. 5.3.2

The application of the Backward-Equivalence-Law-II procedure results in Fig. 5.3.3:

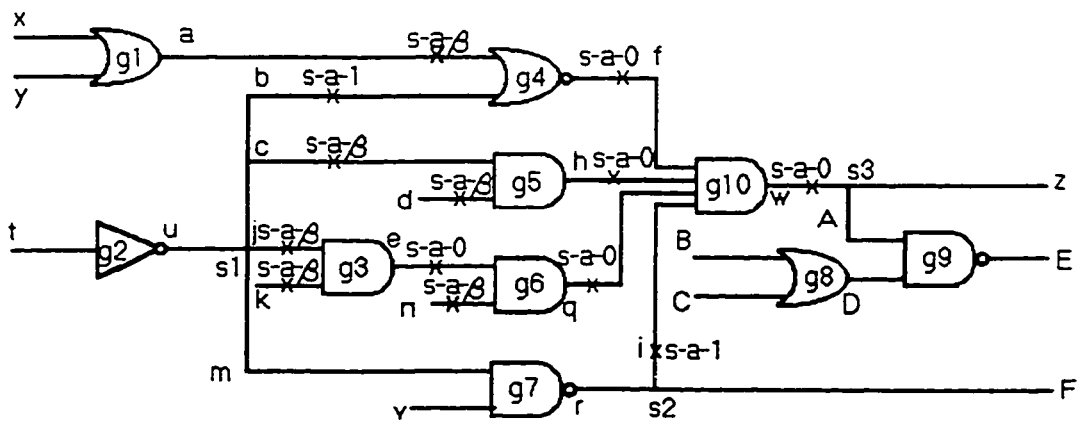


Fig. 5.3.3

Because there is a s-a-0 redundancy at the output of g10, each of the procedures Backward-Equivalence-Law-III, Forward-Equivalence-Law-IV and Backward-Equivalence-Law-IV is applied. The result following the first procedure is shown in Fig. 5.3.4.

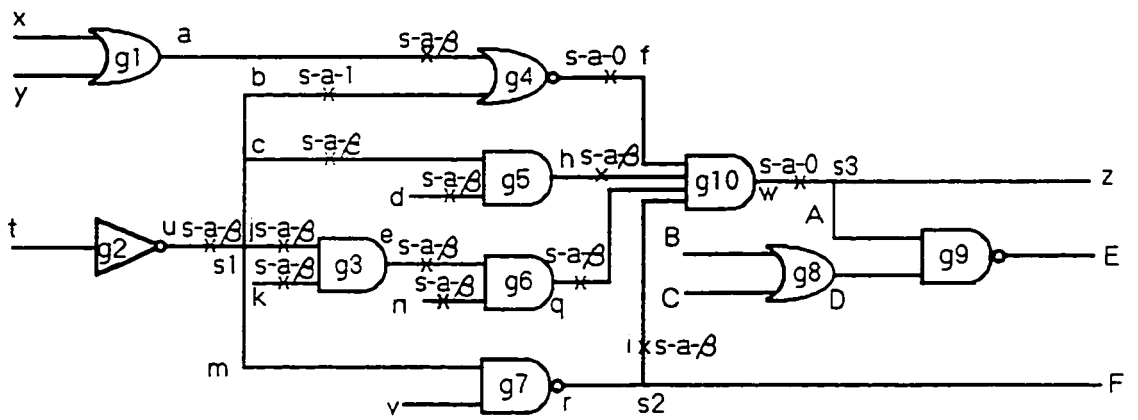


Fig. 5.3.4

The result of applying the Forward-Equivalence-Law-IV procedure is shown in Fig. 5.3.5:

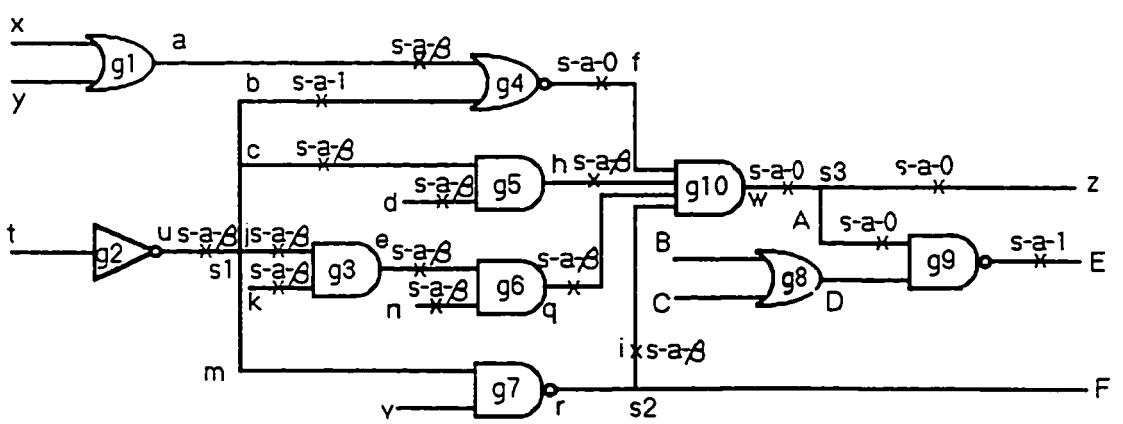


Fig. 5.3.5

The result of applying the Backward-Equivalence-Law-IV procedure is shown in Fig. 5.3.6:

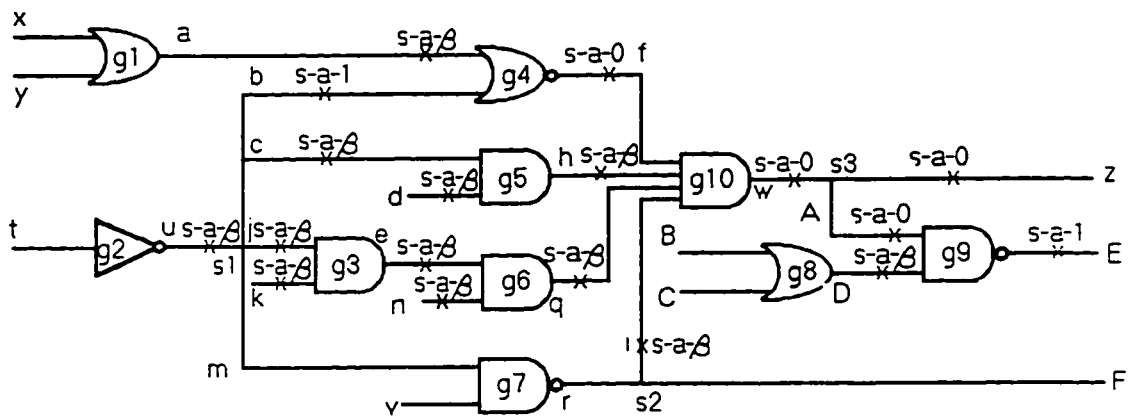


Fig. 5.3.6

Finally the Backward-Equivalence-Law-I procedure is applied and this provides the final result as shown in Fig. 5.3.7:

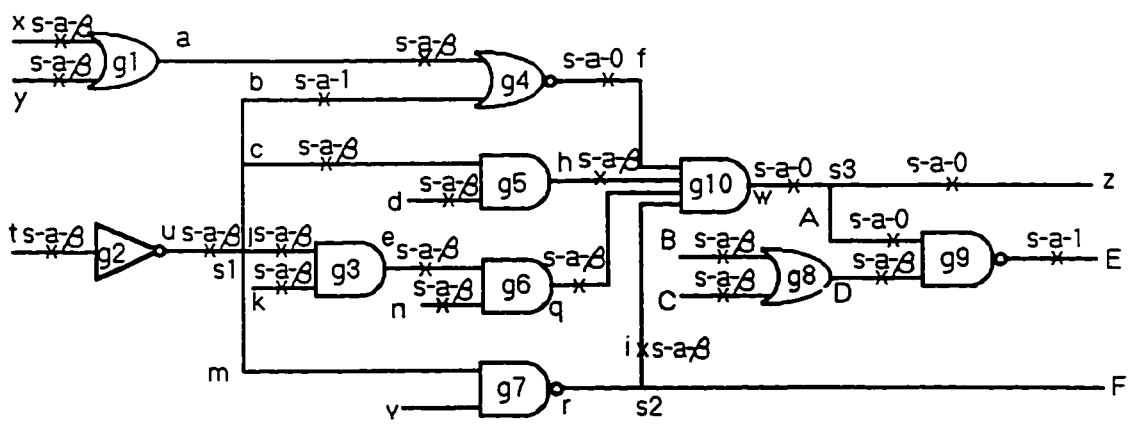


Fig. 5.3.7

The purpose of the Report procedure is to print out the Redundancy-Table which summarizes all redundant faults that have been located. The format of this table is shown below:

Table of Identified Redundancies			
Line-ID	Redundancy	Line-ID	Redundancy
b	s-a-1	c	s-a-β
j	s-a-β	f	s-a-0
h	s-a-0	e	s-a-0
q	s-a-0	i	s-a-1
w	s-a-0	u	s-a-β
z	s-a-0	A	s-a-0
E	s-a-0	D	s-a-β
a	s-a-β	d	s-a-β
k	s-a-β	n	s-a-β
x	s-a-β	y	s-a-β
t	s-a-β	B	s-a-β
C	s-a-β		

Table 5.3.3 Table of Identified Redundancies

Comparisons with HM method:

The ISB method carries out the RI process in two steps; namely, initial redundancy identification and the application of the Equivalence Laws. Each step traverses the circuit once only. The HM method, in contrast, the traverses the circuit more than twice.

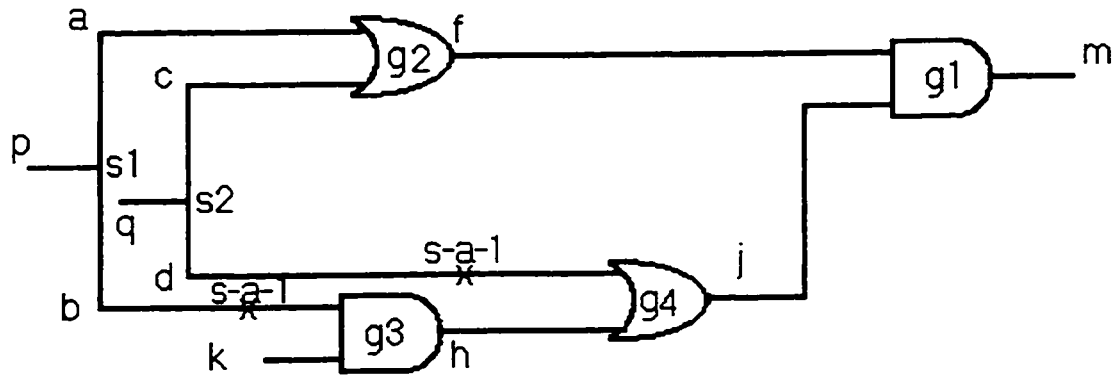
5.4 Multiple-Stem Region Analysis

The nature of multiple stem regions was introduced in section 2.2. Such regions require special treatment and we now outline how the ISB method is modified. In fact, the only difference between RI for single-stem region RI and multiple-stem region RI is the procedure for cv path identification and initial redundancy identification.

In a multiple-stem region of dimension m, each sG-path has multiple beginning lines each of which connects to one of the stemlines. Also, an essential property of the FGL gates in a multiple-stem region is that each such gate is reachable from every one of the stemlines of the region.

The cv path identification and initial redundancy identification for multiple-stem regions is carried out in two steps. These are described below and each is illustrated with an example.

Fig. 5.4.1 shows a such multiple-stem region. There are two sG-paths; namely, P(a, c, f) which has beginning lines a and c, and P(b, d, j) which has beginning lines b and d. The gates g2 and g4 are FGL gates, but g3 is not a FGL gate.



5.4.1

Step-1/5.4 For every sG-path P, if any stemline value combination, v_1, v_2, ..., v_m, when processed over the path P, uniquely determines a value at the output of the reconvergent gate G, then (x_1': s-a-$\bar{v}_1$), (x_2': s-a-$\bar{v}_2$), ..., (x_m': s-a-$\bar{v}_m$) are initial redundant faults on the beginning lines of every sG-path P' which is different from P.

In Fig. 5.4.1, for the sG-path P(a, c, f), the stemline value combination (p = 0, q = 0) implies the value 0 at the output of the reconvergent gate g1. Hence from Step-1/5.4, (b: s-a-1) and (d: s-a-1) are redundant faults.

Step-2/5.4. For each FGL gate, after the application of the Step-1/5.4, check if any FGL gate has s-a-u redundancy on all input lines, where u is the non-controlling value of the gate. If yes, then apply FEL III (see section 4.4).

Consider Fig. 5.4.2. After the application of the Step-1/5.4, a s-a-1 redundancy is identified on all the input lines of g2. Note that 1 is the non controlling value of g2, so FEL III can be applied. As a result, (h: s-a-0) is identified redundant.

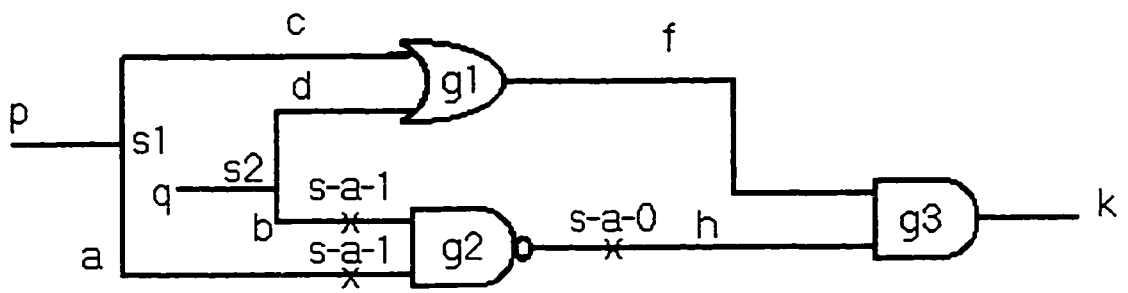


Fig. 5.4.2

Notice that the Step-1/5.4 is equivalent to an analogous procedure as in the Single-Stem-Region process. Step 2, however, is a special case which arises from the multiple-stem structure.

The procedure for initial redundancy identification for the multiple-stem region can now be presented subsequently. The three tables, MS-Exit-Path-Table, MS-Initial-Redundancy-Table and MS-Virtual-Initial-Redundancy-Table are equivalent to the analogous tables introduced for the single-stem region case. The Determinative-Value-Table used in the Exit-Path-Identification procedure has two columns and each row in the table corresponds to an sG-path. The first entry in a row is an sG-path specification. The second entry is the value, u, which if assigned to the output of the FGL gate on the sG-path in question will uniquely determine the value at the output line of the reconvergent gate, G. If no such value exists, then the sG-path is not added to the table. The value u is called the **determinative value** of the sG-path.

PROC. MS-Initial-RI
(R(s1,s2, ..., s_m, G);
MS-Exit-Path-Table,
MS-Initial-Redundancy-Table,
MS-Virtual-Initial-Redundancy-Table)

START

Exit-Path-Identification (R(s_1, s_2, ..., s_m, G);
Determinative-Value-Table,
MS-Exit-Path-Table)

FOR every sG-path P(x_1, x_2, ..., x_m, y)

```
{
    IF the determinant value, u, of the sG-path exists
    {
        FOR each stemline value combination (v1, v2, ...,
        vm) which produces the value, u, at the output
        line of the FGL gate
        {
            FOR each sG-path P'(x1', x2', ..., xm', y')
            different from P(x1, x2, ..., xm, y)
            {
                IF P' has no exit path
                {
                    FOR i = 1 to m
                    {
                        add (xi': s-a-v̄i) into the MS-
                        Initial-Redundancy-Table;
                    } /end of FOR/
                } /end of IF/
```

```
                    IF P' has an exit path
                    {
                        FOR i = 1 to m
                        {
                            add (x_i': s-a-v̄_i) into the MS-
                            Virtual-Initial-Redundancy-
                            Table;
                        } /end of FOR/
                    } /end of IF/
                } /end of FOR/
            } /end of FOR/
        } /end of IF/
    } /end of FOR/
    /- end of Step-1/5.4 -/

    FOR every FGL gate, g'
    {
        IF all input lines of g' have a s-a-v redundant fault,
        where v is the non-controlling value of g'
        {
            Forward-Equivalency-Law-III (g';
                    Initial-Redundancy-Table,
                    Virtual-Initial-Redundancy-Table)
        } /end of IF/
    } /end of FOR/
    /- end of Step-2/5.4 -/

END.
```

```
PROC Exit-Path-Identification (R(s1, s2, ..., sm, G;
                               Determinative-Value-Table,
                               MS-Exit-Path-Table)

START

     FOR every sG-path P'
     {
          let u be the controlling value of the FGL gate, g';
          let Na be the FGL gate and Nb be the successor node to
          Na;

          IF Nb is a stem line point
          {
               add P' and stem-ID to the MS-Exit-Path-Table;
          } /end of IF/

          WHILE the value on the output of Na is the controlling
          value of Nb
          {
               IF Nb = G
               {
                    {
                         add the P' and u to the Determinative-
                         Value-Table;
                    }
```

```
            ELSE
                {
                    Let Na be Nb and then Nc be the successor
                    node to Na;
                    IF Nc is a stem line point
                    {
                        add P' and stem-ID to the MS-Exit-
                        Path-Table;
                    } /end of IF/
                }
            } /end of IF/
        } /end of WHILE/

        WHILE Nb ≠ G
        {
            let Nb to be the successor of Nc;
            IF Nb is a stem line point
            {
                add P' and stem-ID to the MS-Exit-Path-Table;
            } /end of IF/
        } /end of WHILE/
    } /end of FOR/

END.
```

Now the complete RI procedure for multiple-stem region can be presented:

PROC. Multiple-Stem-Region (R(s, G); Redundancy_Table)

START.

```
MS-Initial-RI (R(s, G);
               Exit-Path-Table,
               Initial-Redundancy-Table,
               Virtual-Initial-Redundancy-Table);

FOR each sG-path, P, which has an initial or virtual initial
redundancy
{
     IF P has no exit path
     {
          {
               Forward-Equivalence-Law-I (P,
                         Initial-Redundancy-Table;
                         OP-Redundancy-Table,
                         G-Output-Redundancy);
          }
     ELSE
          {
               Forward-Equivalence-Law-II (P,
                         Virtual-Initial-Redundancy-Table;
                         Exit-Path-Table;
                         OP-Redundancy-Table,
                         G-Output-Redundancy);
          }
     } /end of IF/
} /end of FOR/
```

```
IF (G-Output-Redundancy ≠ ϕ)
{
     Backward-Equivalence-Law-III (G-Output-Redundancy;
                                   Stemline-Redundancy,
                                  OP-Redundancy-Table);

     Forward-Equivalence-Law-IV (G-Output-Redundancy;
                                 Equivalent-Redundancy-Table);

     Backward-Equivalence-Law-IV(Equivalent-Redundancy-
                                 Table;
                                 NR-Redundancy-Table);
} /end of IF/

Backward-Equivalence-Law-II (OP-Redundancy-Table;
                             FP-Redundancy-Table);

Backward-Equivalence-Law-I (FP-Redundancy-Table,
                            NR-Redundancy-Table;
                            X-Equivalent-Redundancy-Table);

Report (Initial-Redundancy-Table, OP-Redundancy-Table,
        G-output-Redundancy, Stemline-Redundancy,
        Equivalent-Redundancy-Table, NR-Redundancy-Table,
        FP-Redundancy-Table, X-Redundancy-Table;
        Redundancy-Table);

END.
```

Comparison with HM method:

For the multiple-stem region case, the ISB method traverses the whole circuit only once to find all exit paths during the initial redundancy identification (this is the same as for the single-stem region case). Another traversal of the whole circuit is required for the application of the equivalence laws. In the HM method the cv path identification process itself may take three traversals of the whole circuit. Furthermore, our ISB method utilizes FEL III in the multiple-stem region case. There is no equivalent action in the HM method.

5.5 Multiple Region RI Method

The RI process discussed thus far with respect to single-stem regions and multiple-stem regions is restricted to a single region. In this section we enlarge the RI process to redundancies arising in multiple regions.

If a path, C(x, y), starting from point x and ending at point y on a sG-path, belongs to more than one region, we refer to C(x, y) as a **common path** for the regions it belongs to. Our discussion in this section focuses on redundancies that can occur on common paths.

Fig. 5.5.1 provides an example. Line c is the common path, which belongs to regions R(s1, g2) and R(s2, g2).

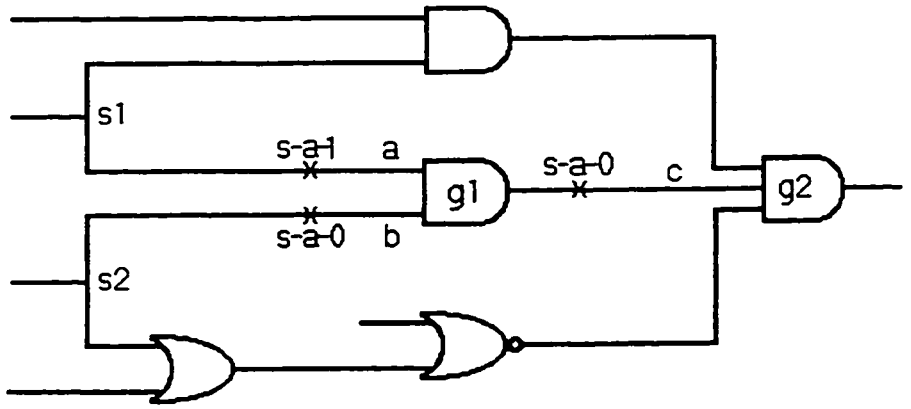


Fig. 5.5.1

The single stuck at fault (c: s-a-0) in Fig. 5.5.1 is not a redundant fault in region R(s1, g2), but is a redundant fault in region R(s2, g2). This illustrates the underlying dilemma. Consequently some rules are required to deal with the RI problem on a common path.

We include, in our ISB method, an approach for solving such a problem; namely, how to identify the existence of redundancies on common paths.

We use Fig. 5.5.2 to introduce our approach.

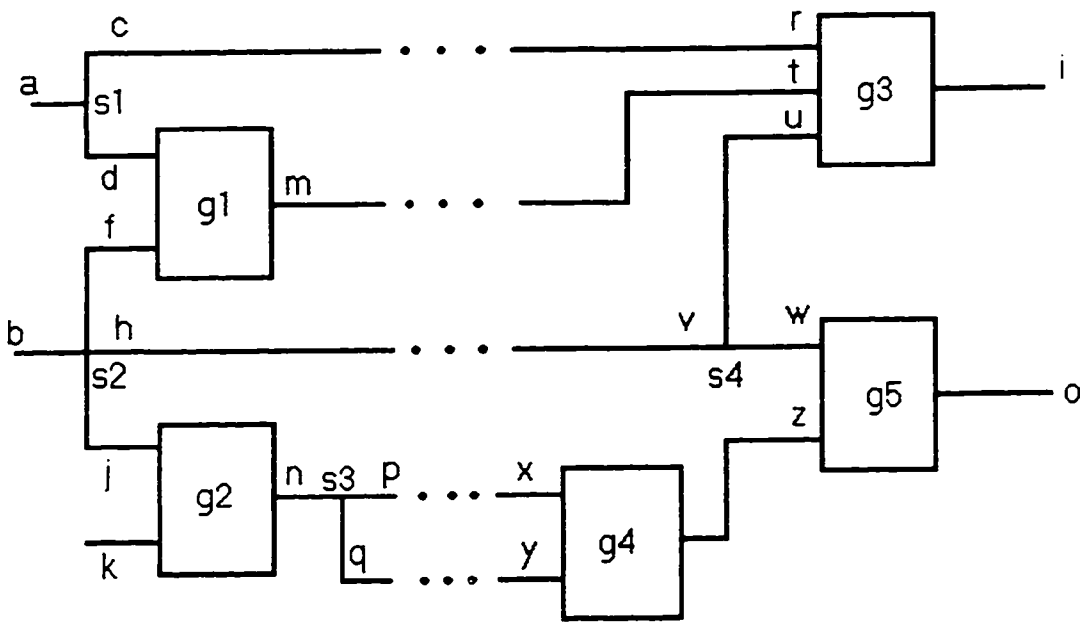


Fig. 5.5.2

Fig. 5.5.2 is a multiple region circuit with four regions: R(s1, g3), R(s2, G3), R(s2, g5) and R(s3, g4). There are four common paths: C(m, t), C(h, v), C(p, x) and C(q, y).

If a region B lies entirely inside another region A, then some sG-paths, P(x, y), in A can include an entire sG-path P'(x', y') in B. A similar observation applies for cv paths in A.

For example, in Fig. 5.5.2, region R(s3, g4) lies entirely within R(s2, g5), and with respect to region R(s2, g5) there are two different sG-paths that begin at line j and end at line z. One includes P(p, x) and the other includes P(q, y), then the sG-path P(j, z) can be a path goes through either P'(p. x) or P'(q, y).

Suppose a multiple circuit has a single region R(s, G) and another single region R(s', G'), and the two single regions have a common path C(x, y). Let L be a line on C(x, y). Now study the following four cases:

In case 1 and case 2, consider G is bound by G'.

case 1 (L: s-a-i) is identified redundant in one single region and is not redundant in another single region.

Because G is bound by G', any single stuck-at fault on L is not redundant only if it can be propagated through G first and then G'. If the propagation is failed in either R(s, G) (i.e., the fault is redundant in R(s, G)) or R(s', G' (i.e., the fault is redundant in R(s', G')), the fault then can be summarized as a redundant fault to the multiple region circuit. Hence in this case, (L: s-a-i) is a redundant fault on the common path C.

case 2 (L: s-a-u) is identified redundant in R(s, G) and (L: s-a-v) is identified redundant in R(s', G').

Using the conclusion in case 1, we know both (L: s-a-u) and (L: s-a-v) are redundant faults to the multiple region circuit. If u = v, then the final result is (L: s-a-u) (or (L: s-a-v)) redundant. If u ≠ v, then the final result is (L: s-a-β) redundant.

Combining the results of the observation of case 1 and case 2, we establish a law as below:

Law-I/5.5: If G and G' are bound by one another, then the redundant faults on a common path L of R(s, G) and R(s', G') is the overlaying of the redundancy on L in each single region; namely, R(s, G) or R(s', G').

In Fig. 5.5.2, for example, if (x: s-a-0) is redundant in R(s3, g4) and (x: s-a-1) is redundant in R(s2, g5), then (m: s-a-β) is redundant to the circuit because g4 is bound by g5 and hence the Law-I/5.5 holds.

In case 3 and case 4, consider G and G' is not bound by one another.

case 3 (L: s-a-i) is identified redundant in one single region and is not redundant in another single region.

Assume (L: s-a-i) is redundant in R(s, G) and not redundant in R(s', G'), then the existence of the R(s', G') is equivalent to an exit path to line L refer to R(s, G) because (L: s-a-i) can be propagated through G'. Hence (L: s-a-i) is not redundant in the multiple region circuit.

case 4 (L: s-a-u) is identified redundant in R(s, G) and (L: s-a-v) is identified redundant in R(s', G').

According to the observation in case 3, R(s', G') can be respected as an exit path to (L: s-a-u) in R(s, G) and symmetrically R(s, G) can be respected as an exit path to (L: s-a-v) in region R(s', G'). Hence, unless u = v; namely, (L: s-a-u) (or (L: s-a-v)) is redundant in both single regions, line L should be redundancy free. If u $\neq$ v, then (L: s-a-u) (or (L: s-a-v)) is a redundant fault to the multiple region circuit.

Combining the results of the observation of case 1 and case 2, we establish a law as below:

Law-II/5.5: If G and G' are not bound by one another, then a line, L, on a common path of the two single regions R(s, G) and R(s', G') has no redundancy unless (L: s-a-i) is identified redundant in both R(s, G) and R(s', G').

For example, in Fig. 5.5.2, if (h: s-a-u) is redundant in region R(s2, g3) but not redundant in region R(s2, g5), it is then not a redundant fault in the circuit since g3 and g5 are not bound by one another.

We finally consider the case where G and G' is the same reconvergent gate for the single regions R(s, G) and R(s', G'). If (L: s-a-i) is redundant in one single region; e.g., in R(s, G), then there must exist a cv path in R(s, G) which is active when (L: s-a-i) is sensitized. So (L: s-a-i) is a redundant fault to the multiple region circuit no matter it is redundant or not in another region, R(s', G'). This conclusion is equivalent to the case G and G' are bound by one another in case 1 and case 2. We can respect G and G' are bound each other when G and G' are the same gate.

Consider the question we presented at the beginning of this section: Is (c: s-a-0) in Fig. 5.5.1 a redundant fault? Now we can say it is because line c is a common path of R(s1, g2) and R(s2, g2), where the reconvergent gate in both single regions are the same gate, and (c: s-a-0) is identified redundant in R(s2, g2).

With the above two laws and the observation for the last case, G = G', we can establish a general idea of the **Multiple Region RI Method.**

Multiple Region RI Method:

a) Determine redundancies in each single region independently.

i) Identify cv paths and initial redundancies in each region independently.

ii) Apply FEL I and FEL II to identify all redundancies on sG-paths.

b) Apply Law-I/5.5 on each common path which belongs to R(s, G) and R(s', G'), where G and G' are bound by one another or G = G'.

c) Apply Law-II/5.5 on each common path which belongs to R(s, G) and R(s', G'), where G is not bound by G'.

d) Apply equivalency laws as appropriate, to identify other equivalent redundant faults.

We use Fig. 5.5.3 to illustrate our Multiple-Region RI method:

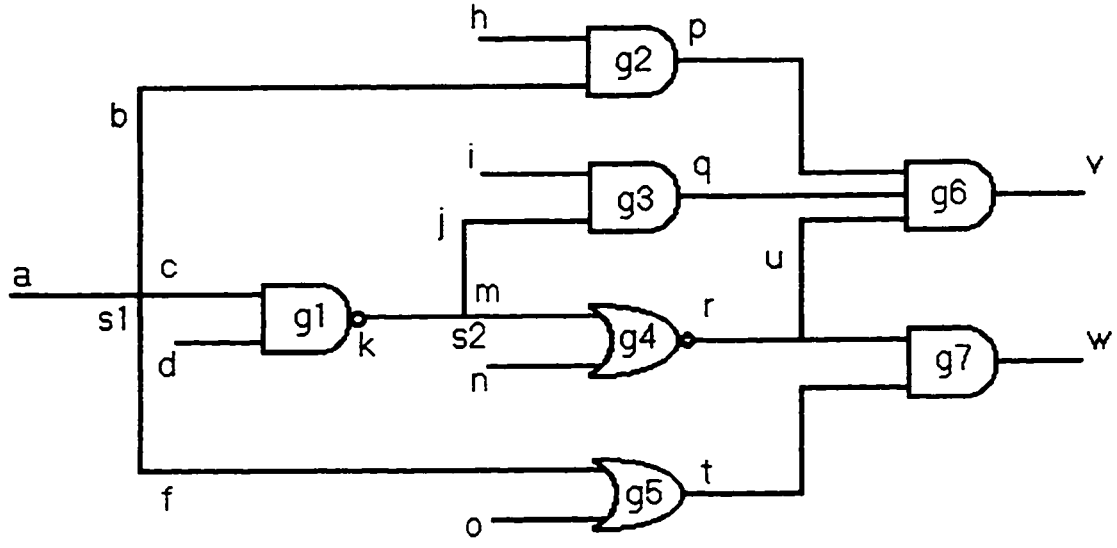


Fig. 5.5.3

There are three reconvergent regions and four common paths in the circuit. The single reconvergent regions are:

. R(s1, g6)
. R(s1, g7)
. R(s2, g6)

The common paths are:

. C(c, k) belongs to R(s1, g6) and R(s1, g7)
. C(j, q) belongs to R(s1, g6) and R(s2, g6)
. C(m, r) belongs to R(s1, g6), R(s1, g7) and R(s2, g6)
. C(u, u) belongs to R(s1, g6) and R(s2, g6)

After step a), the redundancies in the three regions are shown as Fig. 5.5.4(a), 5.5.4(b) and 5.5.4(c):

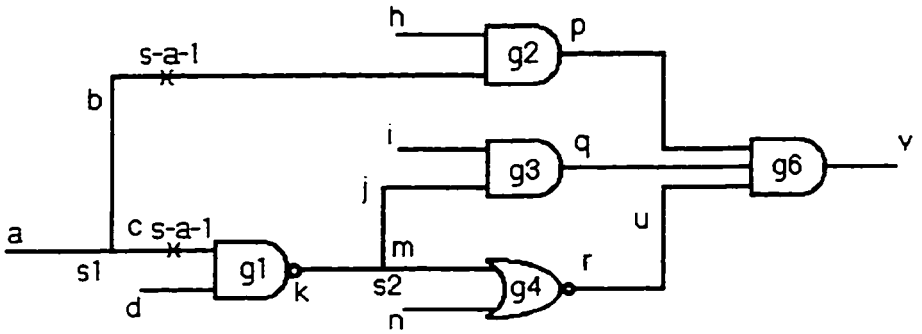


Fig. 5.5.4(a)

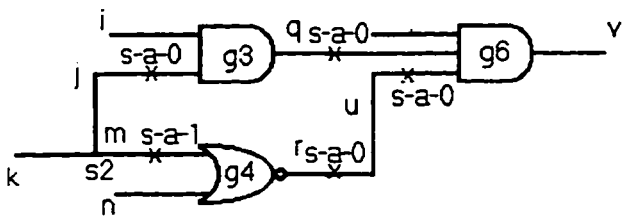


Fig. 5.5.4(b)

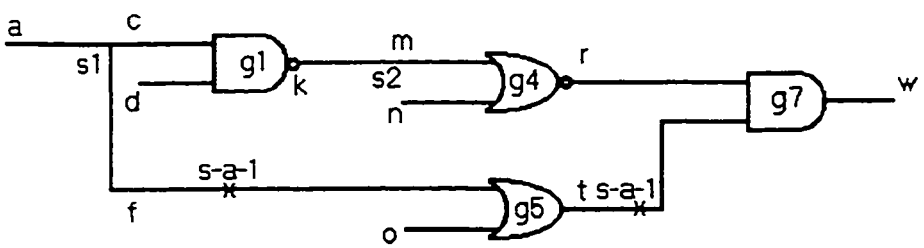


Fig. 5.5.4(c)

According to step b), we overlap the redundancies on C(m, q), C(m, r) and C(u, u) from R(s1, g6) and R(s2, g6), which results in Fig. 5.5.5(a):

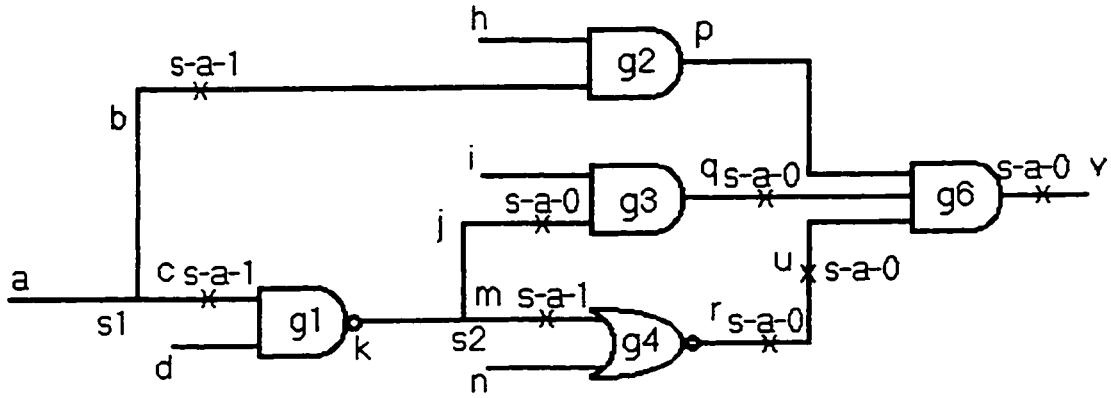


Fig. 5.5.5(a)

Following the application of step c), the redundancies on C(c, k) and C(m, r) are removed because they do not exist in both R(s1, g6) (or R(m, r)) and R(s1, g7). This is shown in Fig. 5.5.5(b):

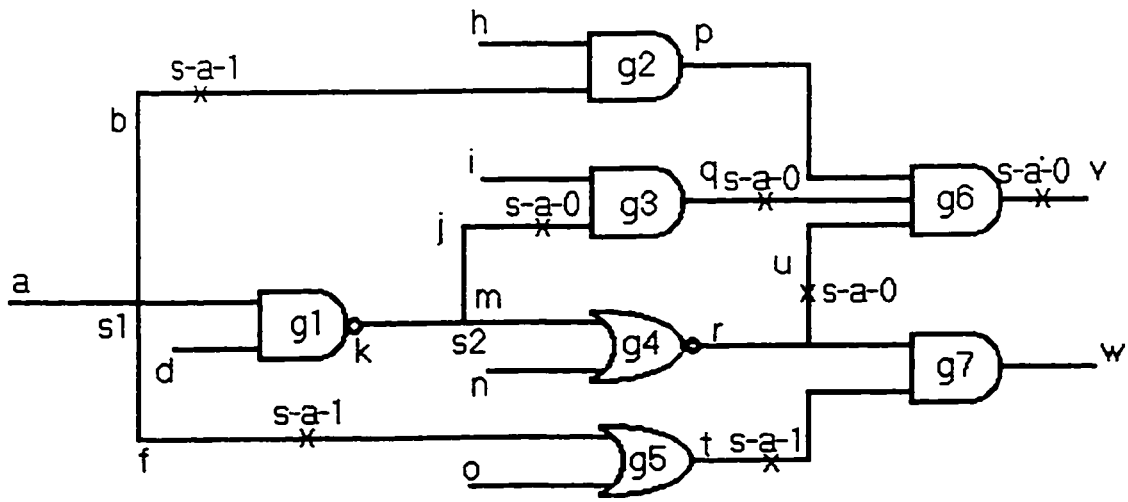


Fig. 5.5.5(b)

Following step d), the final result is shown in Fig. 5.5.6:

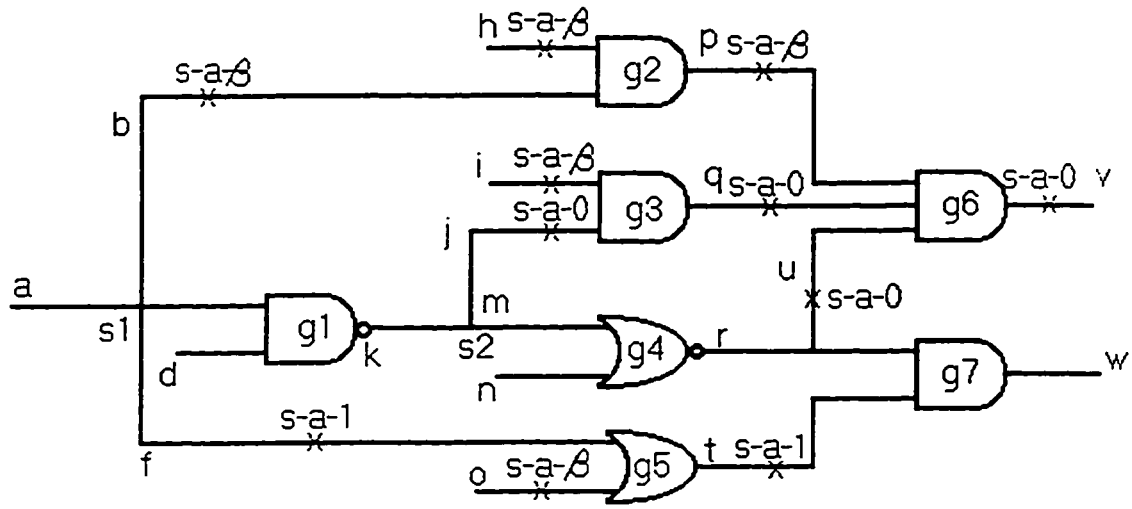


Fig. 5.5.6

Comparison with the HM method:

The common path problem is not treated in the HM method.

5.6 Comparisons with Some Existing Methods

A number of comparisons between our IS method and the HM method have been provided and the HM method have been provided at the end of each of the sections in this chapter. We emphasize here two of the main advantages of the IS method; namely,

a) The IS method makes fewer traversals of the circuit and consequently is able to more efficiently carry out the redundancy identification task.

b) Many of the underlying concepts in the IS method can be applied to circuits with multiple regions, thereby extending the range of applicability of the method.

The most distinctive and powerful feature in the IS method is the set of equivalence laws which have been formulated and incorporated in the method. These provide an exceptionally effective means for uncovering redundant faults once a set of initial redundancies has been identified. Both the initial redundancy identification and the equivalence laws are based on structural features of the circuit.

Some equivalence features are included in both the EST and the DYRID approach (see section 3,4 and section 3.5 respectively). Rule-A/3.4 in EST is an equivalence law. Its weakness, however, is that it cannot be applied backwards. Our backward equivalence laws therefore enhance the capabilities of the IS method relative to the EST approach.

The "test cover" law in DYRID provides a backwards equivalence feature. But it has limited effectiveness in uncovering certain types of equivalent redundancies. In IS method, however, all equivalent redundancies are located. This is primarily due to our careful distinction between on-path and off-path input lines.

The "safe FOB" law (DY2/3.5) relates to the important fact that the Rule-A/3.4 cannot be applied in some special case. This point is carefully studied in Chapter 4 of the thesis where a related law (FEL IV) is then formulated.

We consider now a final example to demonstrate the extended capability of the IS method with respect to the DYRID approach. Figure 5.6.1 is an example problem considered in [2](see also Fig. 3.5.2(b) relating to our earlier summary of DYRID).

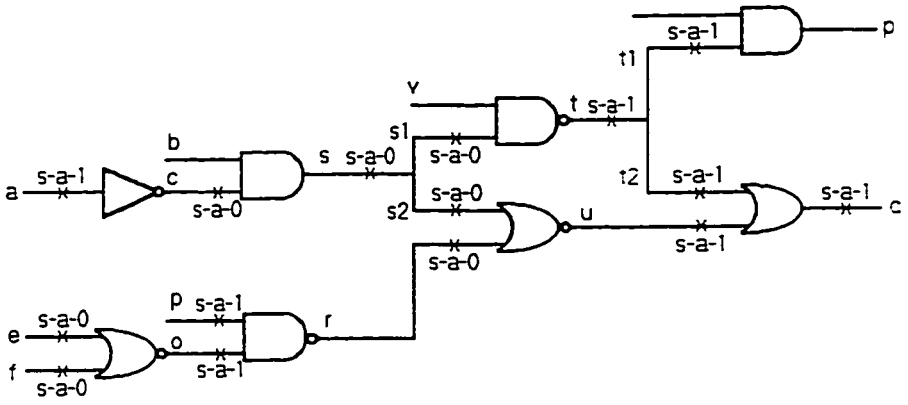


Fig. 5.6.1

Fig. 5.6.1 shows the result of applying DY1/3.5 and DY2/3.5 after (a: s-a-1) is identified redundant. In addition to the equivalent redundancies shown in Fig. 5.6.1, IS method can further identify some other equivalent redundancies as shown in Fig. 5.6.2.

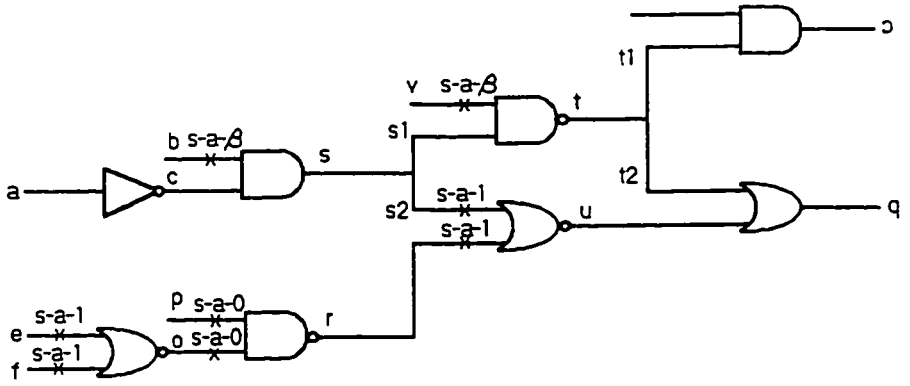


Fig. 5.6.2

CHAPTER VI SUMMARY AND CONCLUSIONS

6.1 Summary

Our concern in this thesis has been with the problem of redundancy (undetectable fault) identification in combinational circuits. Following a brief overview of some existing methods (chapter 3) we focused particular attention on the structure based method of Harihara and Menon[23] (the HM method). Our particular effort has been with extending and improving the various structure-based concepts introduced in the HM method. This has resulted in a new redundancy identification method which we call the ISB (Improved Structure-Based) method.

A main feature of the ISB method is a set of equivalence laws (chapter 4) which we have formulated. These assist in identifying redundant faults with minimal computational overhead and should therefore contribute to a superior structure-based redundancy identification approach. The approach is based on the identification of a set of initial redundancies followed by the application of the equivalence laws. An initial redundancy may occur on the beginning line of each sG-path within a reconvergent region. Furthermore, each is a representative of a family of equivalent redundancies and our equivalence laws provide the basis for identifying members of the family.

In addition we have proposed an alternate means of cv path identification. Our alternative has features which we believe will enhance the effectiveness with which this aspect of the redundancy identification process can be carried out. Some new results relating to multiple-stem regions have also been identified based on the application of our Forward Equivalence Law III.

Some investigation of the special problems arising in multiple regions has also been carried out. Our ISB method therefore is more widely applicable than the HM method.

6.2 Conclusions

As noted by Cormen et al [15], redundancy identification is an NP-complete problem. The redundancy identification process can therefore only be actually carried out as an approximation approach.

No implementation of our ISB method has yet been achieved because of the extensive effort that would be involved in such a task. Consequently our contribution in this thesis does not include experimental results with the benchmark circuits that are typically employed to check the efficiency of redundancy identification procedures. Nevertheless, we have illustrated the effectiveness of most of the key elements of our ISB method using various examples which in some cases have been extracted from benchmark circuits.

Our work is fundamentally an extension of the structure-based concepts introduced by Harihara and Menon[23]. It enhances the efficiency with which structure-based analysis can be carried out. We believe that an implementation of the ISB method would provide a superior tool for redundancy identification.

REFERENCES

[1] M. Abramovici, D. Miller and R. Roy, "Dynamic Redundancy Identification in Automatic Test Generation", IEEE Trans. on CAD, Vol. 11, pp. 404-408, March 1992.

[2] Miron Abramovici, Melvin A. Breuer and Arthur D.Friedman. "Digital System Testing and Testable Design", Computer Science Press, 1990.

[3] M. Abramovici, P. Menon and D. Miller, "Critical Path Tracing: An Alternative to Fault Simulation", IEEE Design and Test, pp. 83-92, Feb. 1984.

[4] V. Agrawal and D. Johnson, "Logic Modeling of Faults", pp. 86-88", IEEE 1986.

[5] S. Akers, "Binary Decision Diagrams", IEEE Trans. on Computers, Vol. c-27, pp. 509-516, June 1978.

[6] S. Akers, B. Krishnamurthy, S. Park and A. "Why Is Less Information from Logic Simulation More Useful in Fault Simulation", Swaminathan, 1990 International Test Conference, pp. 786-800, 1990.

[7] K. Antreich and M. Schulz, "Accelerated Fault Simulation and Fault Grading in Combinational Circuits",IEEE Trans. on CAD, pp. 704-711, Sept. 1987.

[8] K. Antreich and M. Schulz, "Fast Fault Simulation in Combinational Circuits", IEEE Int. Conf. on CAD, pp. 330-333, Nov. 1986.

[9] B. Ayari and B. Kaminska, "A Cyclomatic Testablity Measure", pp. 8.2.1-8.2.10.

[10] D. Brand, "Redundancy and Don't Cares in Logic Synthesis", IEEE Transactions on Computers, Vol. c-32, pp. 947-952, October 1983.

[11] F. Brglez and H. Fujiwara. "A Neutral Netlist of 10 Combinational Benchmark Circuits", Proc. of ISCAS; Special Session on ATPG and Fault Simulation, pp. 151-158, IEEE, June 1985.

[12] F. Brglez, P. Pownall and R. Hum, "Accelerated ATPG and Fault Grading Via Testability Analysis", IEEE Proceeding of ISCAS 85, pp. 695-698. 1985.

[13] D. Bryan, F. Brglez and R. Lisanke, "Redundancy Identification and Removal", MCNC Workshop on Logic Synthesis, pp. 1-14, 1989.

[14] W. Cheng, "The Back Algorithm for Sequential Test Generation", Proc. 1988 IEEE Int. Conf. on Computer Design, pp. 66-69, October 1988.

[15] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, "Introduction to Algorithms", chapter 36.

[16] H. Fujiwara, "Logic Testing and Design for Testability", The MIT Press, 1985.

[17] H. Fujiwara and T. Shimono, "On The Acceleration of Test Generation Algorithms", IEEE Trans. on Computers, Vol. C-32, pp. 1137-1144, Dec. 1983.

[18] J. Giraldi and M. Bushnell, "Search State Equivalence for Redundancy Identification and Test Generation", International Test Conference, pp. 184-193, 1991.

[19] J. Giraldi and M. Bushnell, "EST: The New Frontier in Automatic Test-Pattern Generation", 27th ACM/IEEE Design Automation Conference, pp. 667-672, 1991.

[20] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuit", IEEE Trans. on Computers, Vol. C30, pp. 215-222, March 1991.

[21] A. Goundan and J. Hayes, "Identification of Equivalent Faults in Logic Networks", IEEE Trans. Computers, Vol c-29, pp. 978-985, Nov, 1980.

[22] T. Gruning, U. Mahlstedt, W. Daehn and C. Ozean, "Accelerated Test Pattern Generion by Cone-Oriented Circuit Partitioning", pp. 418-421, 1990.

[23] M. Harihara and P. R. Menon, "Identification of Undetectable Faults in Combinational Circuits", International Conference on Computer Design, pp. 290-293, 1989.

[24] S. Hong, "Fault Simulation Strategy for Combinational Logic Networks", Proc. 8th Int. Symp. on Fault Tolerant Comp., pp. 96-99, June 1978.

[25] S. Kajihara, H. Shiba and K. Kinoshita, "Test Generation and Removal of Redundancy under Classification of Undetectable Faults", pp. 1-10.

[26] W. Ke and S. Seth, "A Fast Fault Simulation Algorithm for Combinational Circuits", Proc. of 8th Int. Conf. on CAD, pp. 166-169, Nov. 1988.

[27] T. Kirkland and M. Mercer" A Topological Search Algorithm for ATPG", 24th ACM/IEEE Design Automation Conference, paper 28.2, pp. 502-508, 1987.

[28] T. Kirkland and M. Mercer, "Algorithms for Automatic Test Patern Generation", IEEE Design and Test of Computers, pp. 43-55, June 1988.

[29] A. Lioy, "Advanced Fault Collapsing", IEEE Design and Test of Computers, pp. 64-71, 1992.

[30] F. Maamari and J. Rajski, "A Method of Fault Simulation Based on Stem Regions", IEEE Trans. on CAD, Vol. 9, pp. 212-220, Feb. 1990.

[31] F. Maamari and J. Rajski, "Reconvergent Fanout Analysis and Fault Simulation Complexity of Combinational Circuits", IEE Electronics Letters, Vol.23, pp. 1131-1133, Oct. 1987.

[32] R. Marlett, "EBT: A Comprehensive Test Generation Technique for Highly Sequential Circuits", Proc. of the 15th Design Automation Conf., pp. 335-339, June 1978.

[33] E. McCluskey and F. Clegg, "Fault Equivalence in Combinational Logic Networks", IEEE Trans. Computers, Vol. c-20, pp. 1286-1293, Nov. 1971.

[34] Meiji University, "Research Reports of the Faculty of Engineering", No 50, 1986.

[35] P. Menon, Y. Levendel and M. Abramovici,"Critical Path Tracing in Sequential Circuits", Int. Conf. on CAD, pp. 162-165, Nov. 1988.

[36] S. Patil and P. Banerjee, "A Parallel Branch and Bound Algorithm for Test Generation", IEEE Trans. on CAD, Vol. 9, pp. 313-322, March 1990.

[37] M. Roberts and P. Lala, "An Algorithm for The Partitioning of Logic Circuits", IEE Proceedings Vol. 131, pp. 113-118, July 1984.

[38] M. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Application to Redundancy Identification", IEEE Transactions n Computer-aided Design. Vol. 8, No. 7, pp. 811-816, July 1989.

[39] M. Schulz and E. Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques", Proc. Int. Symp. Fault Tolerant Comput, pp. 30-35, June 1988.

[40] M. Schulz, E. Trischler and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", IEEE Trans. on CAD, pp. 126-137, 1988.

[41] M. Schulz, E. Trischler and T. Sarfert "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System", Int. Test Conf., pp. 1016-1026, 1987.

[42] M. Schulz and F. Brglez, "Accelerated Transition Fault Simulation", 24th ACM/IEEE Design Automation Conference, paper 15.1, pp. 237-243, 1987.

[43] J. Waicukauski and E. Lindbloom, "Transition Fault Simulation by Parallel Pattern Single Fault Propagation", 1986 International Test Conference, pp. 542-549, 1986.